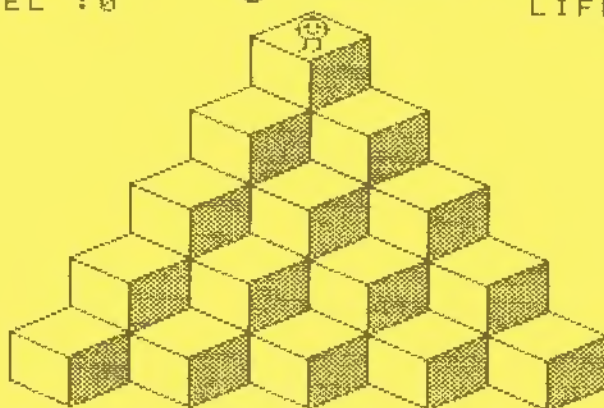
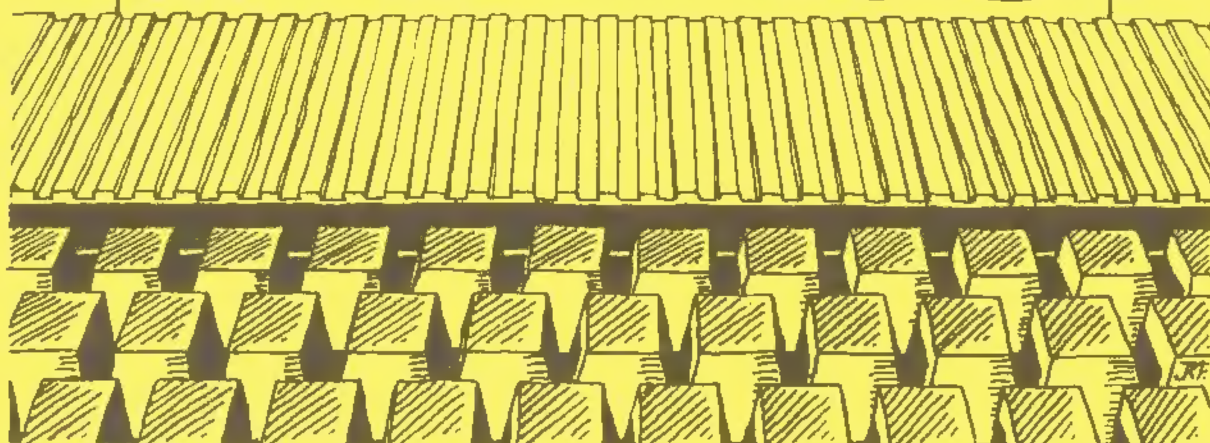
ATOM NIEUWS

JAARGANG
NUMMER





ATOM NIEUWS UIT DE FEDERATIE

Bestuur:

Voorzitter:
N. Stad
Plataanweg 47
1544 PB Zaandijk
Tel. 075-280808

Secretaris:
N. Schoone
Sluispad 55-B
1521 EZ Wormerveer
Tel. 075-286624

Penningmeester:
Th. van Kempen
Het Puyven 71
5672 RB Nuenen
Tel. 040-836210

Clubwinkel:
N. Stad
Plataanweg 47
1544 PB Zaandijk
Tel. 075 - 280808

Hard-ware cie.:
P. Ehrlich
Roostenlaan 266
5644 BS Eindhoven
Tel. 040 - 114183

Redaktie Atom Nieuws:
H. de Ruiter
Polarisstraat 25
8303 AC Emmeloord

Contributie 1986: fl. 60,00
Leden Buitenland: fl. 75,00

Giro 5244293 Bank 52.84.69.010
Beide t.n.v. Atom Computer Club, Nuenen

Redaktie Atom Nieuws:
Joop Ballijns
Harry de Ruiter
Ed Schijf
Evert van Schothorst
Jaap van der Veen
Gerhard Visser
Wibo Visser

Redaktieadres:
Postbus 1373
8001 BJ Zwolle

Ledenadministratie:
S. van Leeuwen
Kompasstraat 32
1973 PX IJmuiden
Tel. 02550 - 22435

Datasheets:
G. Akkermans
Wikke 1
1273 BR Huizen
Tel. 02152 - 60294

Drukwerkarchief:
F. Monsanto
C. Kohlerstraat 97
7558 VC Hengelo

Uiterste datum inlevering copy:
nr. 2 19 - 2 - 1986

DE CLUB-WINKEL:

Geheugenkaart; 16 kByte extra in de ATOM	fl. 40,00
Schakelkaart; meerdere EPROM's op Axxx	fl. 47,50
Minischakelkaart	fl. 16,00
Herdruk ACORN NIEUWS 1982; 97 pag. wetenswaardigheden	fl. 6,00
Jaargang 1983; totaal ruim 450 pag.	fl. 30,00
Jaargang 1984; totaal ruim 650 pag.	fl. 35,00
Jaargang 1985; totaal ruim 650 pag.	fl. 35,00
ATOM-WARE deel 1; machinetaal op de ATOM, 98 pag.	fl. 6,00
ATOM-WARE deel 2; het diskdrive operatingsystem, 68 pag.	fl. 5,00
ATOM-WARE deel 3; het monitor operating system, 80 pag.	fl. 5,00

Levering:
Via uw regionale penningmeester of rechtstreeks bij Atom Computerclub te Nuenen, uitsluitend na voorafgaande betaling (vermeerderd met fl. 4,00 voor portokosten).
Indien u 14 dagen na afschrijving van uw bank- of girorekening nog niets hebt ontvangen, informeer dan bij de clubwinkel.

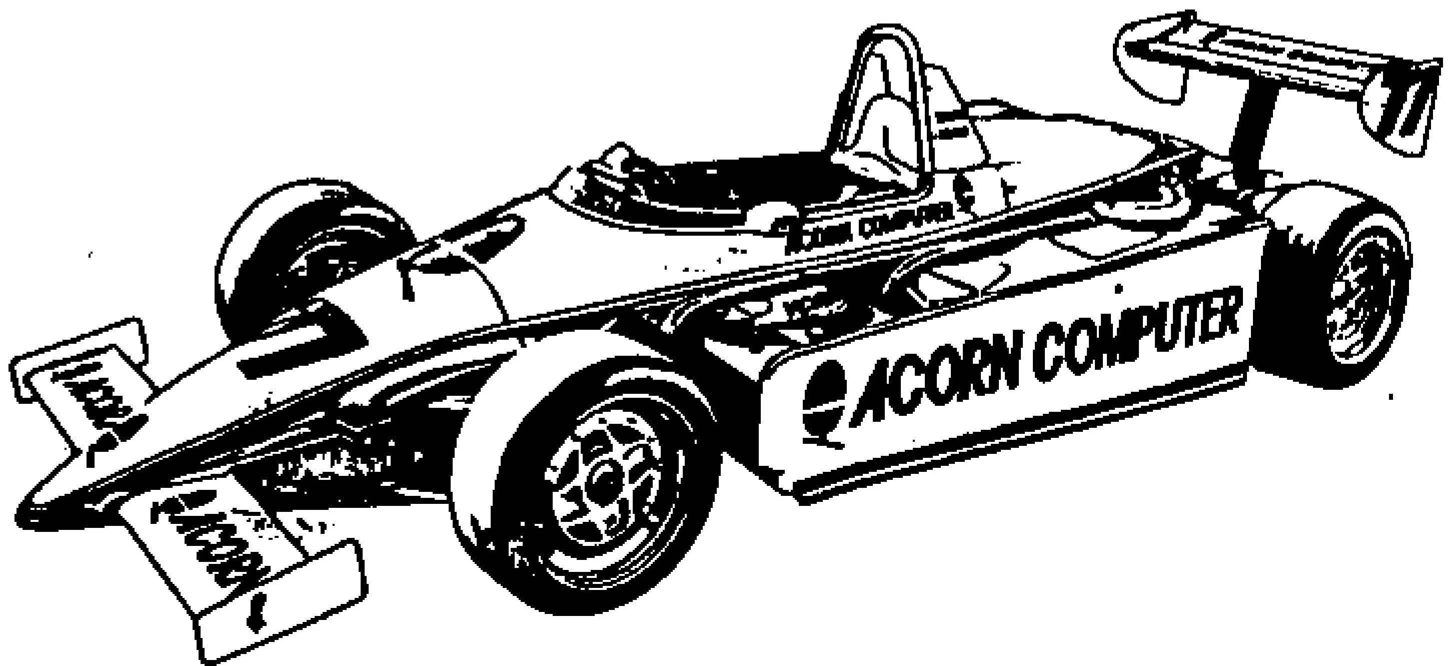
REDACTIONEEL

Een nieuw jaar, een oud jaar, het is een kwestie van komen en gaan.
Vooreerst beste lezers namens de redactie een gelukkig, succesvol en ontspannen 1986 toegewenst!
In de afgelopen tijd is een terugkijken op wat geweest is en wat gebeurt is een normale zaak. Voorop dit nummer ziet u de u wel bekende eikeltjes die aangeven de jaargang en het nummer van het betreffende jaar.
Het zal u opvallen dat het nummer wat u in uw handen hebt nummer 1 van de vijfde jaargang is. Voor een computerclub een gedenkwaardig feit. Een dankwoord voor de "oude" garde is dan ook zeker op zijn plaats.
Betrekken we daarbij de uitgebrachte ontwikkelingen en ontwerpen en de nog uit te brengen zaken dan mogen we vol goede moed een nieuw jaar tegemoed zien.
Zeker als we, gelet op de prestaties, bedenken dat we een HOBBY computerclub zijn.

met vriendelijke groet,

het re(d)actie-team

P.S. Heeft de verantwoordelijke figuur binnen uw regio de gegevens betreffend uw regio (namen,adressen,etc.) reeds naar het federatief bestuur gezonden?



INHOUD REGIOSCHIJF 1-A

C-DATA	8000	8000	02000	002
LOCKSPR	2900	C2B2	01289	022
MON.RH	2900	C2B2	0135A	035
BPROM	2900	C2B2	03EB3	049
TURTCOM	2900	C2B2	0161E	088
UPSDOWN	2900	C2B2	00137	09F
MOVPLOT	2900	C2B2	00364	0A1
INDEXER	2900	C2B2	00428	0A5
INTRA	2900	C2B2	02600	0AA
SPEECH	2900	C2B2	01511	0D0
TRAVEL	2900	C2B2	01D9C	0E6
DEFKEY	2900	C2B2	00DBF	104
HAMMUR	2900	C2B2	00E95	112
CASTLE	2900	CE86	01700	121
GAGS2.2	A000	A000	01000	138
ALLWAYS	2900	C2B2	00581	148
KLOKJE	2900	C2B2	006AF	14E
EDIBUG	6000	A731	01000	155
IMAGE	2900	C2B2	00C3E	165
UNI-PR	2900	C2B2	016FF	172
CTRLIST	2900	C2B2	0013E	189

INHOUD REGIOSCHIJF 1-B

SCHAKEL	2900	C2B2	0052F	00F
BINGO73	2900	C2B2	003D3	015
BINGO90	2900	C2B2	00366	019
VERRASS	2900	C2B2	00667	01D
FREQ	2900	C2B2	00448	024
POGO	2900	C2B2	0136D	029
RADEN	2900	C2B2	00D93	03D
PBUFFER	2900	C2B2	005F9	04B

Op de regioschijf treft u nogmaals GAGS aan. Het betreft hier de versie 2.2 aangezien er nog enkele aanpassingen mogelijk bleken te zijn die dit geheel nog gebruikersvriendelijker maakten dan ze al was. Het betreft een aanpassing in de Sprite-handling en CLS werkt via write vector niet meer rechtstreeks in de monitor rom.

TRAVEL

Het programma TRAVEL is voorlopig het laatste uit de reeks van operations-research programma's (zie voor andere toepassingen PERT uit Acorn Nieuws 1983 no.5 blz 49, SIMPLEX uit Acorn Nieuws 1985 no.2 blz 28, en JOB-SHOP uit De Cursor 3e jrg no.3).

TRAVEL levert de oplossing voor zog. reistijden problemen.Een mooi voorbeeld is dat van de handelsreiziger, die een 12-tal steden moet bezoeken en die graag de kortste route langs al deze steden wil weten ten einde zo min mogelijk reistijd (en benzine) kwijt te zijn.

Het vinden van de kortste route is geen eenvoudig probleempje als je je realiseert dat er bijna 40 miljoen verschillende manieren zijn om deze 12 steden te bezoeken.En, helaas, al deze 40 miljoen mogelijkheden zullen geevalueerd moeten worden om de juiste oplossing (= kortste weg) te vinden: er bestaat geen trucje om dit grote getal tot een wat werkzamer kleiner getal om te vormen.
Het bijgevoegde programma checkt alle routes op een hele slimme manier: als ergens halverwege de evaluatie van een bepaalde route blijkt dat de af te leggen afstand al groter is dan de tot dan toe gevonden kortste weg dan beeindigt het programma de evaluatie van die route, inclusief alle variaties op die route.Als je precies wilt weten hoe dat gebeurt, lees dan eens het artikel van de heren Parry&Pfeffer in Byte van juli 1981 blz 252, waar alle details beschreven staan.Indien nodig kan ik voor een copie zorgen.

Dat deze heren behoorlijk in hun opzet geslaagd zijn, blijkt wel uit de snelheid van het programma: een 12-bezoeken probleem wordt door onze Atom in precies een uur opgelost en dat is echt geen geringe prestatie voor een in Basic geschreven stukje tekst.Het eveneens bijgevoegde 10-bezoeken probleem wordt in ongeveer 23.5 minuten opgelost (er moeten in dit geval 40320 alternatieven nagetrokken worden.Je ziet, als het aantal te evalueren routes met een factor 1000 toeneemt, neemt gelukkig de benodigde rekentijd slechts met een factor van minder dan drie toe).

Centraal in het programma staat de "interdestination-table" die de onderlinge afstand tussen al de te bezoeken plaatsen bevat.Deze tabel wordt berekend uitgaande van de afstand en de windrichting t.o.v een referentiepunt (overigens hoeft dit referentie punt geen deel uit te maken van de te bezoeken plaatsen).In het eerste voorbeeld wordt deze windrichting d.m.v. poolcoordinaten ingevoerd, in het tweede voorbeeld eenvoudig d.m.v. zuid,noord etc.N.B. de windrichting wordt aangegeven vanuit het referentie punt naar het op te geven punt toe. Dus als het referentie punt Utrecht is en Arnhem deel uit maakt van de route, dan is de windrichting oost.

Uitgebreide editing faciliteiten zijn beschikbaar voor eenmaal ingevoerde data (zie vooral het tweede voorbeeld).Een opmerking nog: als de interdestination tabel te groot wordt dan past hij

niet netjes meer op het scherm (dit is het geval in het eerste voorbeeld).Je zult dan eventueel de print layout wat aan moeten passen.

De op deze wijze gemaakte interdestination-tabel geeft de hemelsbrede afstanden (dus zoals een vogel zou vliegen).In de praktijk kan een voertuig niet altijd een rechte weg tussen twee punten afleggen.Loopt de route langs grote steden, die vaak door autowegen met elkaar verbonden zijn, dan is de hemelsbrede afstand vrijwel gelijk aan de echte.In een stad echter, of over kronkelige dijken, dan klopt een en ander niet meer.Daarom bestaat de mogelijkheid de tabel te editen.

In het tweede voorbeeld worden daarom eerst ruwweg de afstanden ingevoerd m.b.v. de windrichting.Daarna worden de berekende afstanden die al te erg afwijken vervangen door de echte (de interdestination tabel wordt in integer afgedrukt; het programma rekent echter met de niet afgeronde werkelijke FP-afstanden).
Daarnaast is het ook mogelijk de tabel direct te vullen.

Verder vraagt het programma om de plaats, waar je beginnen wilt en om de plaats, waar je eindigen wilt. Beide plaatsen kunnen identiek zijn (voorbeeld 1), maar dat hoeft niet (voorbeeld 2).Dit tweede geval is vooral handig als je perse een plaats als laatste wilt bezoeken (het is niet zo handig de babysit op je boodschappen rit mee te moeten nemen).

Heb je nog vragen,bellen staat vrij.
Peter Ruifrok
v.d.Duyn v. Maasdamstr.46
5344 HS Oss
tel.04120-30581

****EERSTE VOORBEELD****

```
the travelling salesman

HOW MANY DESTINATIONS?12


TYPE 1 IF YOU WANT TO INPUT THE
ANGLE AS MAP DIRECTIONS
(E.G. SW,NNW,E,ENE ETC.)

TYPE 2 IF YOU WANT TO INPUT THE
ANGLE USING POLAR COORDINATES
(E.G. 0 DEGREES=EAST, 90 DEGREES=NORTH, 180 DEGREES=WEST ETC)

TYPE 3 IF YOU WISH TO INPUT ALL
DATA VIA THE INTER-DESTINATION
TABLE

METHOD 1,2 OR 3?2

1.NAME OF DESTINATION
(MAX 10 CHARACTERS)?PEORIA
```

```
DISTANCE FROM REFERENCE?128

ANGLE(Ø DEGREES IS EAST)?223


2.NAME OF DESTINATION
(MAX 1Ø CHARACTERS)?CHICAGO

DISTANCE FROM REFERENCE?Ø

ANGLE(Ø DEGREES IS EAST)?Ø


3.NAME OF DESTINATION
(MAX 1Ø CHARACTERS)?BELLEVILLE

DISTANCE FROM REFERENCE?261

ANGLE(Ø DEGREES IS EAST)?244


4.NAME OF DESTINATION
(MAX 1Ø CHARACTERS)?CARBONDALE

DISTANCE FROM REFERENCE?297

ANGLE(Ø DEGREES IS EAST)?255


5.NAME OF DESTINATION
(MAX 1Ø CHARACTERS)?ROCKFORD

DISTANCE FROM REFERENCE?7Ø

ANGLE(Ø DEGREES IS EAST)?163


6.NAME OF DESTINATION
(MAX 1Ø CHARACTERS)?DECATUR

DISTANCE FROM REFERENCE?158

ANGLE(Ø DEGREES IS EAST)?247


7.NAME OF DESTINATION
(MAX 1Ø CHARACTERS)?WAUKEGAN

DISTANCE FROM REFERENCE?27

ANGLE(Ø DEGREES IS EAST)?1Ø4


8.NAME OF DESTINATION
(MAX 1Ø CHARACTERS)?CHAMPAIGN

DISTANCE FROM REFERENCE?126

ANGLE(Ø DEGREES IS EAST)?261


9.NAME OF DESTINATION
(MAX 1Ø CHARACTERS)?DEKALB

DISTANCE FROM REFERENCE?58

ANGLE(Ø DEGREES IS EAST)?184


1Ø.NAME OF DESTINATION
(MAX 1Ø CHARACTERS)?SPRINGFILD

DISTANCE FROM REFERENCE?178

ANGLE(Ø DEGREES IS EAST)?238


11.NAME OF DESTINATION
(MAX 1Ø CHARACTERS)?KANKANEE

DISTANCE FROM REFERENCE?59

ANGLE(Ø DEGREES IS EAST)?266


12.NAME OF DESTINATION
(MAX 1Ø CHARACTERS)?AURORA

DISTANCE FROM REFERENCE?34

ANGLE(Ø DEGREES IS EAST)?2Ø4


***INPUT DATA TO BE USED***

DESTINATION     DISTANCE    BEARING
1.PEORIA          128.Ø       223
2.CHICAGO           Ø.Ø         Ø
3.BELLEVILLE      261.Ø       244
4.CARBONDALE      297.Ø       255
5.ROCKFORD         7Ø.Ø       163
6.DECATUR         158.Ø       247
7.WAUKEGAN         27.Ø       1Ø4
8.CHAMPAIGN       126.Ø       261
9.DEKALB           58.Ø       184
1Ø.SPRINGFILD     178.Ø       238
11.KANKAKEE        59.Ø       266
12.AURORA          34.Ø       2Ø4

DO YOU WISH TO EDIT ANY (Y/N)?N

wait
DO YOU WANT TO EDIT OR EXAMINE
THE INTERDESTINATION-TABLE?N
```

```
WHAT IS YOUR BEGINNING LOCATION?1
WHAT IS YOUR ENDING LOCATION?1


wait
I MUST EVALUATE 39916800
POSSIBILITIES
wait


***THE SHORTEST TRIP***

1.PEORIA       90.6031779
2.DEKALB       26.1405199
3.ROCKFORD     60.6807679
4.WAUKEGAN     27.0000000
5.CHICAGO      34.0000000
6.AURORA       52.4736080
7.KANKAKEE     67.4208960
8.CHAMPAIGN    46.9755790
9.DECATUR     142.247534
10.CARBONDALE  64.3770955
11.BELLEVILLE  86.0116470
12.SPRINGFILD  63.6607532
13.PEORIA       0.0

THE SHORTEST TRIP IS
761.591578



****TWEEDE VOORBEELD****

the travelling salesman

HOW MANY DESTINATIONS?10


TYPE 1 IF YOU WANT TO INPUT THE
ANGLE AS MAP DIRECTIONS
(E.G. SW,NNW,E,ENE ETC.)

TYPE 2 IF YOU WANT TO INPUT THE
ANGLE USING POLAR COORDINATES
(E.G. 0 DEGREES=EAST, 90 DEGREES=NORTH, 180 DEGREES=WEST ETC)


TYPE 3 IF YOU WISH TO INPUT ALL
DATA VIA THE INTER-DESTINATION
TABLE

METHOD 1,2 OR 3?1

1.NAME OF DESTINATION
(MAX 10 CHARACTERS)?THUIS

DISTANCE FROM REFERENCE?0
```

REGIO NOORD

ALGEMENE LEDENVERGADERING

20 februari 1986 om 19.30
in de TREFKOEL,
Zonnelaan 30, Groningen

AGENDA:

1. Opening
2. Ingekomen stukken
3. Verslag secretaris
4. Verslag penningmeester
5. Begroting voor 1986
6. Verslag kascommissie
7. Kandidaatstelling:
   bestuur en kascommisie
8. Pauze
9. Verkiezing
   bestuur/kascommissie
10. Rondvraag
11. Sluiting

MAP HEADING?E

2.NAME OF DESTINATION
(MAX 10 CHARACTERS)?BOEKHANDEL

DISTANCE FROM REFERENCE?3

MAP HEADING?NE

3.NAME OF DESTINATION
(MAX 10 CHARACTERS)?KAASBOER

DISTANCE FROM REFERENCE?4

MAP HEADING?N

4.NAME OF DESTINATION
(MAX 10 CHARACTERS)?SCHOOL

DISTANCE FROM REFERENCE?10

MAP HEADING?NW

5.NAME OF DESTINATION
(MAX 10 CHARACTERS)?GARAGE

DISTANCE FROM REFERENCE?4

MAP HEADING?NW

6.NAME OF DESTINATION
(MAX 10 CHARACTERS)?SUPERMARKT

DISTANCE FROM REFERENCE?5

MAP HEADING?W

7.NAME OF DESTINATION
(MAX 10 CHARACTERS)?BANK

DISTANCE FROM REFERENCE?6

MAP HEADING?SSW

8.NAME OF DESTINATION
(MAX 10 CHARACTERS)?BAKKER

DISTANCE FROM REFERENCE?3

MAP HEADING?S



Now all we have to do is miniaturise ourselves!

```
A=#8000;DOB=A?31;F.X=30TO0S.-1;A?(X+1)=A?X;N.;?A=B;U.0
```

```
A=#8000;DOB=?A;F.X=1TO31;A?(X-1)=A?X;N.;A?31=B;U.0
```

```
9.NAME OF DESTINATION
(MAX 10 CHARACTERS)?DROGIST

DISTANCE FROM REFERENCE?5

MAP HEADING?S


10.NAME OF DESTINATION
(MAX 10 CHARACTERS)?BABYSIT

DISTANCE FROM REFERENCE?4

MAP HEADING?SE


***INPUT DATA TO BE USED***

DESTINATION     DISTANCE    BEARING
1.THUIS           0.0         E
2.BOEKHANDEL      3.0         NE
3.KAASBOER        4.0         N
4.SCHOOL         10.0         NW
5.GARAGE          4.0         NW
6.SUPERMARKT      5.0         W
7.BANK            6.0         SSW
8.BAKKER          3.0         S
9.DROGIST         5.0         S
10.BABYSIT        4.0         SE

DO YOU WISH TO EDIT ANY (Y/N)?Y

TYPE 0 TO END EDITING WHEN ASKED'WHICH ONE'

WHICH ONE?4


4.NAME OF DESTINATION
(MAX 10 CHARACTERS)?SCHOOL

DISTANCE FROM REFERENCE?1

MAP HEADING?NW

WHICH ONE?7


7.NAME OF DESTINATION
(MAX 10 CHARACTERS)?BANK

DISTANCE FROM REFERENCE?6

MAP HEADING?SW

WHICH ONE?0
```

```
DOIN.A,B;C=A;D=B;DOD=D+(C>D)&1*B;C=C+(C<D)&1*A;U.C=D;P.C';U.0
```

INPUT DATA TO BE USED

DESTINATION	DISTANCE	BEARING
1.THUIS	0.0	E
2.BOEKHANDEL	3.0	NE
3.KAASBOER	4.0	N
4.SCHOOL	1.0	NW
5.GARAGE	4.0	NW
6.SUPERMARKT	5.0	W
7.BANK	6.0	SW
8.BAKKER	3.0	S
9.DROGIST	5.0	S
10.BABYSIT	4.0	SE

DO YOU WISH TO EDIT ANY (Y/N)?N

wait
DO YOU WANT TO EDIT OR EXAMINE
THE INTERDESTINATION-TABLE?Y

INTERDESTINATION TABLE
(VALUES ROUNDED)

	1	2	3	4	5	6	7	8	9	10
1.	0	3	4	1	4	5	6	3	5	4
2.	3	0	3	3	5	7	9	6	7	5
3.	4	3	0	3	3	6	9	7	9	7
4.	1	3	3	0	3	4	6	4	6	5
5.	4	5	3	3	0	4	7	6	8	8
6.	5	7	6	4	4	0	4	6	7	8
7.	6	9	9	6	7	4	0	4	4	7
8.	3	6	7	4	6	6	4	0	2	3
9.	5	7	9	6	8	7	4	2	0	4
10.	4	5	7	5	8	8	7	3	4	0

DO YOU WISH TO EDIT ANY(Y/N)?Y

TO ALTER, USE FORMAT:
FROM,TO,NEW DISTANCE
FOR EXAMPLE, 2,4,512 ALTERS
THE DISTANCE FROM DESTINATION 2
TO DESTINATION 4 TO 512.
DISTANCE FROM DESTINATION 4
TO DESTINATION 2 IS ALSO CHANGED
INPUT 0,0,0 TO LEAVE EDIT MODE

1.FROM?1
TO?4
DISTANCE?4.2

2.FROM?2
TO?3
DISTANCE?5.4

3.FROM?4
TO?5
DISTANCE?5.1

4.FROM?5
TO?6
DISTANCE?6.1

5.FROM?8
TO?9
DISTANCE?5.1

6.FROM?8
TO?10
DISTANCE?4.5

7.FROM?9
TO?10
DISTANCE?5.4

8.FROM?3
TO?5
DISTANCE?5.5

9.FROM?6
TO?7
DISTANCE?7.1

10.FROM?5
TO?6
DISTANCE?6.6

11.FROM?0
TO?0
DISTANCE?0

DO YOU WANT TO EDIT OR EXAMINE
THE INTERDESTINATION-TABLE?Y

INTERDESTINATION TABLE
(VALUES ROUNDED)

	1	2	3	4	5	6	7	8	9	10
1.	0	3	4	4	4	5	6	3	5	4
2.	3	0	5	3	5	7	9	6	7	5
3.	4	5	0	3	6	6	9	7	9	7
4.	4	3	3	0	5	4	6	4	6	5
5.	4	5	6	5	0	7	7	6	8	8
6.	5	7	6	4	7	0	7	6	7	8
7.	6	9	9	6	7	7	0	4	4	7
8.	3	6	7	4	6	6	4	0	5	5
9.	5	7	9	6	8	7	4	5	0	5
10.	4	5	7	5	8	8	7	5	5	0

```
DO YOU WISH TO EDIT ANY(Y/N)?N

WHAT IS YOUR BEGINNING LOCATION?1
WHAT IS YOUR ENDING LOCATION?10


wait
I MUST EVALUATE 40320
POSSIBILITIES
wait


***THE SHORTEST TRIP***

1.THUIS          3.00000000
2.BOEKHANDEL     5.02089946
3.GARAGE         5.50000000
4.KAASBOER       3.37531113
5.SCHOOL         4.34367295
6.SUPERMARKT     5.83095131
7.BAKKER         4.42087695
8.BANK           4.30970861
9.DROGIST        5.40000000
10.BABYSIT       0.0


THE SHORTEST TRIP IS
41.2014204
```

```
DOIN.A,B;DOC=A%B;A=B;B=C;U.C=0;P.A';U.0
```

****

DE INTERPRETER ZOEKT STATEMENT. GODFRIED DOLS

****

Om misverstanden te voorkomen, dit is geen advertentie, maar een duidelijke uitleg (vooral voor de beginner) van een moeilijk iets.

Sinds kort ben ik zo af en toe bezig met het proberen te begrijpen van de ATOM-monitor (ROM). Een mooie aanzet hiervoor gaf het boekje ATOM-WARE, deel 1 (Hoofdstuk Basic interpreter).
Voor iemand die van toeten noch blazen weet als het erom gaat hoe de interpreter werkt is het nog een hele kluif dit uit te zoeken.
Nu, ik heb een begin gemaakt en het lijkt mij goed hier iets over te vertellen. Ongetwijfeld zijn er veel leden die van toeten en blazen weten, maar volgens mij zijn er meer die van "niets" weten. Aldus enige eenvoudige uitleg over het zoeken naar een al of niet bekend statement. Voor iemand die het even niet meer kan volgen, is het nuttig in het stroomschema van blz. te kijken.

Allereerst is het nuttig te weten dat een statement waarnaar gezocht wordt, wordt verdeeld in twee stukken, nl.: het eerste is het eerste karakter van het statement en het tweede deel is de rest.
Voor het statement GOTO is het eerste deel karakter "G" en het tweede deel van het karakter "OTO". Nu is er een tabel gemaakt met alle mogelijke beginkarakters van alle statements en een tabel met alle tweede delen van statements. Komt de interpreter in een programma het statement GOTO tegen, dan zal eerst gezocht worden in de eerste tabel of het eerste karakter "G" gevonden wordt. Zo ja, dan wordt gekeken of het volgende karakter misschien een punt is. Het statement zou dan afgekort zijn. Is dit niet het geval (geen afkorting) dan gaat de interpreter verder zoeken in de tweede tabel en vindt daar de karakters "OTO". De interpreter heeft nu het hele statement herkend en springt naar het stukje monitorprogramma welke het statement uitvoert.
Om te begrijpen hoe de interpreter van de ene tabel naar de juiste plaats in de andere tabel springt en van daaruit (als alles goed is) naar het "statement-uitvoerprogramma", moeten we wat meer weten over hoe een tabel in elkaar zit.

Voor de afhandeling van het statement GOTO komen we op adres #C07E in tabel 1 binnen. (Voor tabel zie volgende bladzijde). Daar staat het getal #4C wat karakter "L" voorstelt. De karakters "G" en "L" worden vergeleken en blijken niet aan elkaar gelijk te zijn. Er wordt nu op het volgende adres gekeken. Dit gaat zo door totdat de interpreter op adres #C082 karakter "G" vindt. Bij vergelijking wordt besloten dat beide karakters gelijk zijn.

ADRES #	INHOUD V. ADRES #	WAT STELT HET VOOR
C07D	4E	N
C07E	4C	L
C07F	55	U
C080	4E	N
C081	49	I
C082	47	G
C083	52	R
C084	46	F
C085	21	!
C086	3F	?
C087	24	$
C088	50	P
C089	44	D
C08A	4C	L
C08B	53	S
C08C	42	B
C08D	2A	*

Waarom nu wordt op adres #C07E in de tabel gesprongen (in dit geval)? Welnu, in de interpreter wordt voor het aanwijzen van sprongadressen gewerkt met twee basisadressen #BFFF en #C0EE welke, verhoogd met een variabele index, het werkelijke sprongadres geven.
Zo geeft de interpreter, vlak voor het begin van het zoeken naar een statement bijvoorbeeld GOTO, de index (verhoging t.o.v. een van de basisadressen) #7F mee.
Als we deze index toevoegen (= optellen) aan adres #BFFF, levert dat adres #C07E op. En zie, daar komen we binnen in tabel 1. Nu weet de interpreter vanaf welk adres gezocht moet worden.
Even later wordt op adres #C082 karakter "G" gevonden.
Als we nu terugrekenen naar adres #BFFF (#C082 - #BFFF) blijkt dat de index dan #83 is.
Ziezo, het zoeken naar het eerste deel van het statement is voltooid. Over dus naar de rest (OTO).

Het tweede deel.
De interpreter telt de huidige index #83 op bij basisadres #C0EE, en dit levert adres #C171.
Op adres #C171 staat het getal #BD. De interpreter telt bij dit getal nog een op en het wordt dan #BE. Dit nu is de nieuwe index voor het eerste basisadres #BFFF. Opgeteld leveren deze twee het adres #C0BD op.
En zie nogmaals; op adres #C0BD begint de rest van het statement.

ADRES #	INHOUD V. ADRES	WAT STELT HET VOOR
C0BD	4F	O
C0BE	54	T
C0BF	4F	O
C0C0	CC	
C0C1	45	E
C0C2	54	T
C0C3	55	U
C0C4	52	R
C0C5	4E	N

Na de laatste O van OTO wordt het getal #CC gevonden en de interpreter besluit dat, ten eerste het gehele statement gevonden is en ten tweede dat getal #CC geen karakter is.
Wat moet dat getal dan op die plek?
Welnu, in het begin is verteld dat na het vinden van een statement gesprongen wordt naar een adres waar de uitvoering van dat statement begint.

Ha, nu denken de niet-weters natuurlijk dat #CC wwer een index is voor een van de basisadressen. Fout dus. Getal #CC is een deel van het sprongadres zelf en wel het hoge byte. (Voor niet-kenners: van adres #BFFF is #BF het hoge en #FF het lage byte).
Dit byte bewaren we een tijdje en wel op zeropage-adres #0053.
Even vergeten dus.

Hoe komen we nu aan het lage byte van het sprongadres? We gaan weer met index werken. Toen we klaar waren met het zoeken naar het statement vonden we op adres #COCO getal #CC. Op dat moment was de index #COCO - #BFFF = #C1.
Voegen we deze index toe aan het basisadres #COEE, dan krijgen we adres #C1AF, welke als inhoud getal #05 heeft.
En ja hoor, dit is het lage byte van het sprongadres. We bewaren dit even op zeropage-adres #0052.
Nu wordt een jump JMP(#0052) uitgevoed en dit houdt in dat gesprongen wordt naar het adres welk opgeborgen staat op de adressen #0052 en #0053. Het adres is de samenvoeging van het eerder bewaarde hoge en lage byte, zijnde #CC05.
Op adres #CC05 begint de uitvoering van het statement GOTO.
En dan te bedenken dat dit hele gelees, bewaar en gespring wordt afgehandeld in enkele us.

Ziezo, en hiermee een eind aan dit relaas. Enerzijds omdat we anders door de bomen het bos niet meer zien, anderzijds omdat mijn kennis op een beetje na ook uitgeput is. Maar we gaan stug door.
Nog niet verteld is wat er gebeurt als het statement afgekort is. En wat te doen als het fout gaat en een statement niet herkend wordt?
Bij deze beloofd. Ook hierover komt nog een verhaaltje.

Godfried Dols.

SOFTWARE:One line Bram Poot

```
DOIN.Z;DOP.&Z,&?Z,?Z,"    "$?Z*(?Z>#20)';Z=Z+1;U.?#B001+1;U.O
```

SOFTWARE:Revised TOOLBUG Bram Poot

Inleiding.

De Toolbug is een softwareuitbreiding voor de Atom geschreven door de engelse firma PSION. Deze biedt de gebruiker een aantal programmeerhulpmiddelen op twee verschillende en van elkaar losstaande terreinen. Ten eerste bevat de Toolbug een basic-editor (EDIT) en ten tweede een machinetaal-debugger (DEBUG). Een beschrijving annex handleiding van de Toolbug is verschenen in AcornNieuws 3.1 p.53 e.v., zodat ik hier niet hoef uit te weiden over de aanwezige mogelijkheden.
Dat de Toolbug weinig bekendheid geniet, vindt z'n oorzaak in 2 redenen:

1. de utilities zijn bedoeld voor direct mode en worden dus niet in programma's gebruikt c.q. verspreid (over RENUM en FIND hoor je ook nooit iets).
2. er zitten enkele irritante foutjes in, waardoor sommigen hem al gauw terzijde hebben gelegd.

Ik bedacht me, dat als ik de fouten zou verbeteren, ik er een op sommige punten redelijk geavanceerd softwarepakket bij zou hebben. Ik heb dus de Toolbug en z'n "dislist" uit de kast gepakt en de eerste weer in de schakelkaart geprikt. De hernieuwde belangstelling heeft het volgende opgeleverd.

Ongedocumenteerde instructies.

Hiervan is er slechts 1 en heeft de vorm EDn, waarmee je n regels vanaf de "current line" kunt "extracten" en tegelijkertijd "deleten".
De commandostring L.200^E4^L.200^D4 kan dus verkort worden tot L.200^ED4.
Helaas bestaat de instructie EDM (nog?) niet.

Fouten en hun verbeteringen.

1. De Toolbug-interpreter herkende het commando EDITOR ook als EDIT. Hij testte niet op het einde van het commando. De commando's kunnen nu bovendien niet meer in een programma worden aangeroepen.
2. Het commando L.n waarbij n een niet-bestaand regelnummer is, listte een ogenschijnlijk willekeurig (groot) aantal regels van het programma. De nieuwe versie volgt de beschrijving in de handleiding.
3. Het commando DM wiste precies 1 karakter te weinig in het met M1 en M2 gedefinieerde gebied. Uit de dislist bleek overduidelijk dat en waar een gedachtenfout was gemaakt.
4. In de dump-routine van DEBUG werden de 7-bits ASCII-karakters groter dan #1F afgedrukt INCLUSIEF de DEL (#7F). Een bekende en veelgemaakte fout.

Uitbreidingen en veranderingen.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Een logisch gevolg van al het gezoek en gespit in de kit is het veranderen van dingen zodanig dat ze wat meer voor de hand liggen en/of makkelijker te onthouden zijn. Waarom bv. heet het find-commando H (van hunt) en niet gewoon F? De eerste verandering dus.
Om dezelfde reden heb ik de commando's B (bottom) en T (top) omgewisseld, dus T (top) en B (begin). Top breng ik in verband met het eind van het programma (P.&TOP) en niet met het begin.
Een nadeel is nu dat B= veranderd is in T=.
De bij de orginele top behorende string **TOP** is veranderd in *BEGIN*.

EDIT is vervangen door EDX t.g.v. EDIT van de word pack.

Het commando ? dat informatie over het programma en de buffer afdrukt en dat ook met OLD is te verkrijgen, heb ik veranderd in @. Deze jumpt naar #2800, zodat een daargeplaatste routine (eindigende met RTS) gebruikt kan worden voor extra's naar keuze.
Als je bv. zou willen weten hoe vaak een bepaalde string in een programma voorkomt, dan tik je eerst deze twee regels in:

```
P=#2800;[SEC;LDA@255;SBC9;STA#16;LDA@0;]
[STA#25;STA#34;STA#43;JSR#C589;JSR#FFED;RTS;]
```

waarna je in EDX de volgende commandostring intikt:

```
?B^F.string.255^@
```

en het antwoord verschijnt in beeld.

De listing na L. kun je onderbreken met ESCape.

Uit gesprekken met voormalige Toolbug-gebruikers bleek dat de melding TOO LONG, RETYPE verwarring stichtte. De nieuwe versie meldt in een dergelijk geval FULL, HIT CR en wacht dan net zolang tot de RETURN-toets wordt beroerd, waarna de regel zoals hij daar staat wordt geinterpreteerd; geen misverstanden meer.
De melding komt nu overigens pas na 128 i.p.v. 64 ingetikte karakters.

De regels van het basic-programma mogen tot 128 karakters lang zijn (was 64).

Voor de DOSsers (maar misschien ook wel voor de COSsers) een extraatje. Met Q<cr> verlaat je de Toolbug op de bekende wijze, maar als je na Q een naam opgeeft, wordt het programma eerst onder die naam geSAVEd voordat je terugkeert naar direct mode.
Bv. QSTITCH. Het executieadres is #C2B2; is eventueel te veranderen in #AFAF of om het even welk ander adres.

Tot zover de veranderingen in EDX. Ook in DEBUG was reden tot "ontevredenheid"

C (van crack (?)) is L geworden (van list) naar analogie van de monitor.RH
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

En verder is M nogal gewijzigd:
1. als de inhoud van een byte een afdrukbaar ASCII-karakter bevat, wordt dat ook als zodanig kenbaar gemaakt.
2. de na M te geven antwoorden spatie en return zijn vervangen door resp. return en Q (van quit). Dat werkt prettiger.
3. als extra antwoord na M is de ^ geintroduceerd, waarmee je achterwaarts het geheugen kunt doorlopen.
4. als je de inhoud van een adres verandert, wordt ter verificatie dit adres met z'n huidige inhoud nogmaals afgedrukt.

Zero page gebruik.

De opmerking dat de Toolbug gebruik maakt van de zero page adressen #08 t.e.m. #0C en #70 t.e.m. #7F is helemaal juist (vreemd dat je daar tegenwoordig niet meer zonder meer vanuit gaat). De indeling staat er echter niet bij en zal ik dus hier geven.

EDX

#08	delimiter van strings in F, C en S
#09	"herhalingsbyte"
#0A,#0B	TOP van de buffer
#0C	high byte van startadres van de buffer
#72,#73	adres van "current line"
#7A,#7B	waarde van decimale parameter
#7C,#7D	adres van M1
#7E,#7F	adres van M2

DEBUG

#08	FORMAT-parameter
#0A,#0B	adres van break point
#0C	originele inhoud van break point
#72,#73	referentie adres (commando W)
#79	opslag van stack pointer
#7A,#7B	eindadres in dump en crack (nu: list)
#7C	A-image
#7D	X-image
#7E	Y-image
#7F	P-image

De niet vermelde adressen worden voor tijdelijke pointers en opslag gebruikt.

Tot slot.

Ik kan me goed voorstellen dat er mensen zijn die zeggen: "De Toolbug is wel leuk, maar........." en vult u de cliche's maar in. Bedenk dan wel dat echt interessante mogelijkheden, vooral als daar wat intelligentie bij komt kijken, niet in een 4k eprom passen. De Toolbug levert, mede gezien z'n lengte, heel aardige faciliteiten waarvan EXTRACT/INSERT de meest waardevolle is naast z'n niet-overschrijvende-AUTO en z'n niet-meer-dan-nodig-RENUMBER.
En met een beetje schakelsoft vallen de Toolbug en de P-charme-editor heel goed te combineren, zodat ze elkaar uitstekend kunnen aanvullen.

- ATOM 1.8 AUTOSYNC -

Dat de ATOM op 1,8 MHz of zelfs 2 MHz kan werken zal de meesten wel bekend zijn.
Naast de aantrekkelijke verhoging van de snelheid biedt 1,8 MHz ook nog de mogelijkheid om storingvrij te kunnen plotten.
(dit kan overigens ook op een andere manier; zie A.N.3).
Voor de tape-routine en andere tijdkritische programma's moet naar 1 MHz teruggeschakeld kunnen worden.
Het lastige van omschakelen is dat de timing van de processor even verstoord kan zijn waardoor de ATOM hangt.
Bovendien moet voor storingvrij plotten herhaaldelijk heen en weer geschakeld worden om de juiste fase te vinden,
wat weer een grote kans op vastlopen geeft. Heb je de juiste fase te pakken dan geldt dat niet eens voor alle graphics modes.
Ik heb een schakelingetje gemaakt wat deze bezwaren verhelpt.

Het principe.

Bij het omschakelen kunnen achter de schakelaar in het ongunstigste geval 3 fase-overgangen snel achter elkaar optreden.
Dit signaal gaat nog wel door een deler (IC 44) maar het kloksignaal voor de processor (phi-in) heeft dan nog
een te korte periode voor goede werking met onvoorspelbaar gedrag als gevolg.
De oplossing is om tussen het uitschakelen van de oude klok en het inschakelen van de nieuwe een korte pause te zetten.
Daarbij moet de deler bij het uitschakelen passief blijven (opgaande flank).

Om storingvrij te kunnen plotten mag de processor niet in het grafisch geheugen bezig zijn als de videogenerator
data inklokt. Een zekere periode ervoor moet ook vrij zijn i.v.m. de toegangstijd van de RAM's.
Wanneer de processor eventueel in het grafisch geheugen duikt is te zien aan het phi-2 kloksignaal (pen 39).
De videogenerator heeft een dergelijk signaal niet. Het ligt voor de hand dat de videogenerator de data inleest
direct voordat een nieuw adres op de bus komt.
Kijken we dus naar een adreslijn van de videogen. b.v. Bo (de laagste bit verandert het meest) dan zal een periode
van ongeveer 200 nS voor een overgang vrij moeten zijn dus phi-2 moet dan laag zijn.
In figuur 1 zijn de mogelijke faserelaties aangegeven tussen Bo en phi-in op 1,8 Mhz (pen 10 van IC 44).
Deze zijn met metingen vastgesteld.
Het kleinste verschil is een halve periode van de 3,6 MHz (ong. 140 nS) wat nogal merkwaardig is: kennelijk wordt de 3,6 MHz klok in de videogen.
soms omgekeerd voordat hij gedeeld wordt.
Het bovenste phi-in signaal is het goede. Het lijkt te vroeg maar het moet nog door de processor

en nog wat IC's waardoor het juist achter de Bo-flank komt zodat gedurende een halve periode (280 nS)
voor de Bo-flank nooit gelezen/geschreven wordt in het grafisch geheugen.
Het phi-in signaal moet dus op een kwart periode nauwkeurig ingesteld worden. Hiervoor zijn 2 mogelijkheden:

-1 Het 3,6 MHz signaal eventueel herhaaldelijk omkeren als de fase niet deugt.
Bezwaar: dit geeft verkorte perioden met kans op vastlopen.

-2 Het 3,6 MHz signaal omzetten naar 7,2 MHz en eventueel herhaaldelijk een hele periode blokkeren als de fase niet deugt.
Dit is toegepast.

Het onderscheid tussen goed en kwaad (signaal) kan getrokken worden door gedurende een periode van ongeveer 130 nS
na de Bo-flank (sample) de phi-in in de gaten te houden; is of wordt hij gedurende die periode laag dan deugt hij niet.

Wanneer de processor grafisch leest/schrijft stoort hij het "normale" Bo signaal van de videogenerator.
In zo'n periode mag dus geen fase-kontrole worden gedaan.

De schakeling.

Gezien de complexiteit van de funktie is de schakeling redelijk eenvoudig.
Er zijn maar 3 kleine TTL-IC's nogig.

De poorten (1),(2) en (3) vormen een wissel. De weerstand,condensator en diode aan pen 5 van (1)
zorgen voor een snel blokkeren en vertraagd vrijgeven bij overschakelen.
Poort (4) en (5) maken van de blokgolf van 3,6 MHz korte pulsen met een frekwentie van 7,2 MHz.
Een D-type flip-flop (6) maakt van positief-gaande flanken van Bo,
negatieve pulsen (sample) die in poort (7) vergeleken worden met het phi-in signaal.
Als het fout is wordt de tweede fifo (9) via poort (8) in de foutstand gezet.
Hij blokkeert dan poort (2). Door de eerstvolgende 7,2 MHz puls wordt hij weer teruggeklokt
zodat 1 puls tegengehouden is. De pennen 2 van (8) en (9) zijn met de omschakelaar om in de 1 MHz stand
de poort (2) permanent te blokkeren via fifo (9). Tevens geeft dit een vertraagd inschakelen.
Poort (10) blokkeert fifo (6) en daarmee het vergelijken als de processor grafisch leest of schrijft.

De bouw.

Om het nabouwen te vereenvoudigen heb ik een printje ontworpen

(figuur 3).
De timing lijkt misschien kritisch maar dat valt mee.
De meest kritische tijd is de pulsduur van fifo (6) : 130 nS.
Hierin zit een tolerantie van minstens 30 procent.
Eventueel voor de C van 180 pF een andere waarde proberen.
Bij mij en anderen werkte het overigens meteen goed.
Pen 13 van IC 44 moet uitgebogen worden. (Een alternatief is om pen 10 uit te buigen en een extra LS393 te stapelen; de onderste is dan voor de cassette-interface).
De print kan het beste boven IC 44 en 45 gemonteerd worden; als niet al te dunne draden gebruikt worden is het geheel zelfdragend.
De voeding kan b.v. afgetapt worden van C13.
In plaats van de schakelaar kan ook een bit van een poort (latch) gebruikt worden, maar dan wel een die nergens anders voor gebruikt wordt. Ik geef de voorkeur aan een schakelaar omdat dan in ieder programma kan worden omgeschakeld.
Ik heb nog nooit van vastlopen last gehad.
De schakeling werkt goed met de dynamische RAM kaart van Telec.
Wel moest de C van 150 pF kleiner (33 pF) gemaakt worden omdat de refresh-cycle anders te lang duurde.
Mijn ATOM kan de hogere klokfrekwentie zonder aanpassing probleemloos aan.
Er schijnen echter ATOM's te zijn die wel een aanpassing nodig hebben.
Zie hiervoor A.N. jaargang 3 nr. 5 blz. 68.

Peter Ehrlich

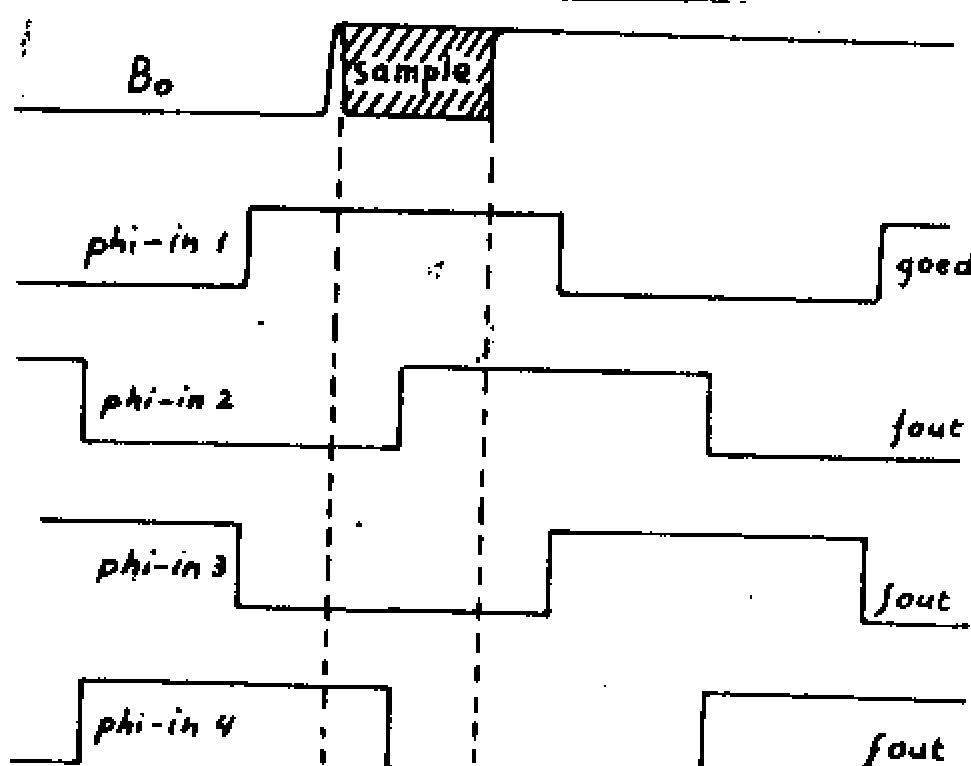


fig. 1

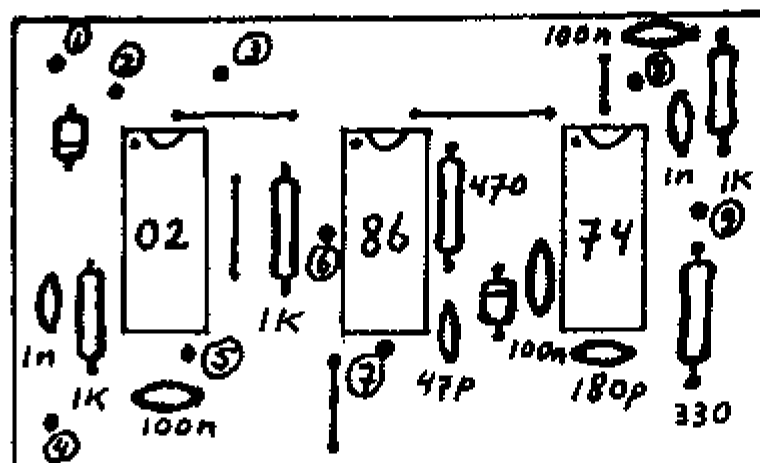


FIG. 3

1. klok uit
2. schakelaar
3. +5V
4. 4 MHz in
5. phi-in
6. 3,6 MHz in
7. VDG in
8. +5V
9. Bo in



LET OP! SCHAAL NIET 1:1

1,2,3,7 : LS02
4,5,8,10 : LS86
6,9 : LS74
diodes: 1N4148 o.i.d.

klok-uit

4 MHz ic 45 p. 1

3,6 MHz ic 9 p. 10

VDG ic 28 p. 1

Bo ic 28 p. 11

phi-in

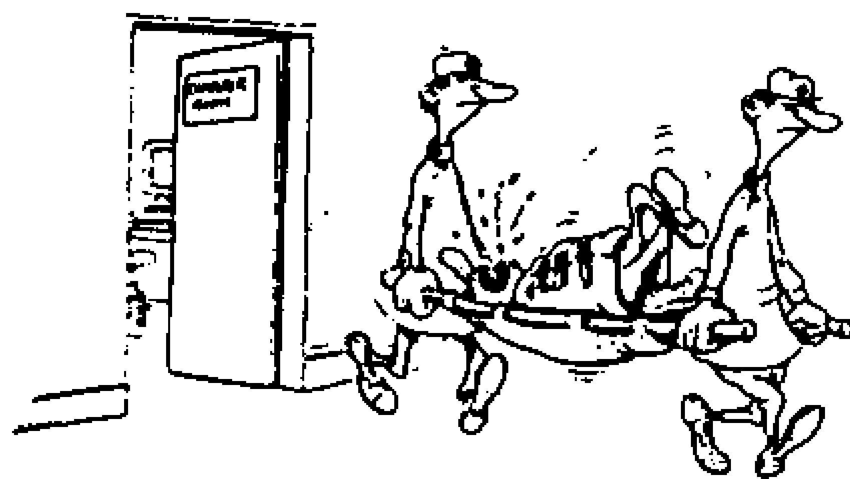
FIG. 2

Een plotter aan de ATOM.

Degene die elektuur lezen zal misschien de advertentie over een zelfbouw plotter zijn opgevallen.
Een prijs van f 345,- was behoorlijk goedkoop, dus maar 1 aangeschaft.
Alles behalve de 24 volts voeding , de tekenplaat en de pennen wordt geleverd, je hoeft alleen maar gaten te boren en de zaak in elkaar te solderen.
Maar dan komt het, hoe bestuur je dat ding. Ik hoor iedereen al zeggen, 'vanuit de ATOM'.
Ja maar dat kost geheugen, vooral als je ook letters wilt verwerken.
Dus waarom niet zoals met printers en diskdrives, via een interface of nog mooier een processor print met RAM en ROM.
Wij aan het puzzelen en ja daar kwamen de ideeen.
Een R65C02-processor op een print met 2 stuks 27128 en een 6821 PIA voor het I/O gebeuren. De 4 adressen voor de 6821 worden van de 2 ROM's afgehaald en dient voor de communicatie met de ATOM en de controle en besturing van de plotter.
Een tweede print werd uitgedacht met 32 K-RAM (6264), eventueel met batterie-backup, welke als een sandwich op de processor print geplaatst wordt. Deze RAM geeft dan de mogelijkheid van een input buffer, maar ook wordt de mogelijkheid van het downloaden van andere dan in ROM geplaatste karakters en figuren gecreeerd.
Dit de hardware, dan de software.
De plotter heeft een oplossend vermogen van 0,04 mm en kan ruim A-3 papierformaat aan.
Waar haal je een behoorlijk nauwkeurige teken routine vandaan, Nou ja uit de ATOM toch.
De ATOM heeft een goede plot routine voor elke soort lijn. Daarnaast komt in de ROM een karakterset, teken routines voor cirkels en elipsen en de mogelijkheid om karakters in allerlei posities te printen.
Daarnaast wordt een hoeveelheid control-codes ingebouwd die onderstrepen, dik tekenen, pen(kleur, 3 totaal) wisseling toelaat en andere mogelijkheden die ook op matrix printers voorkomen zullen ingebouwd worden.

Als er meer te melden is, dan laten we dat weten.

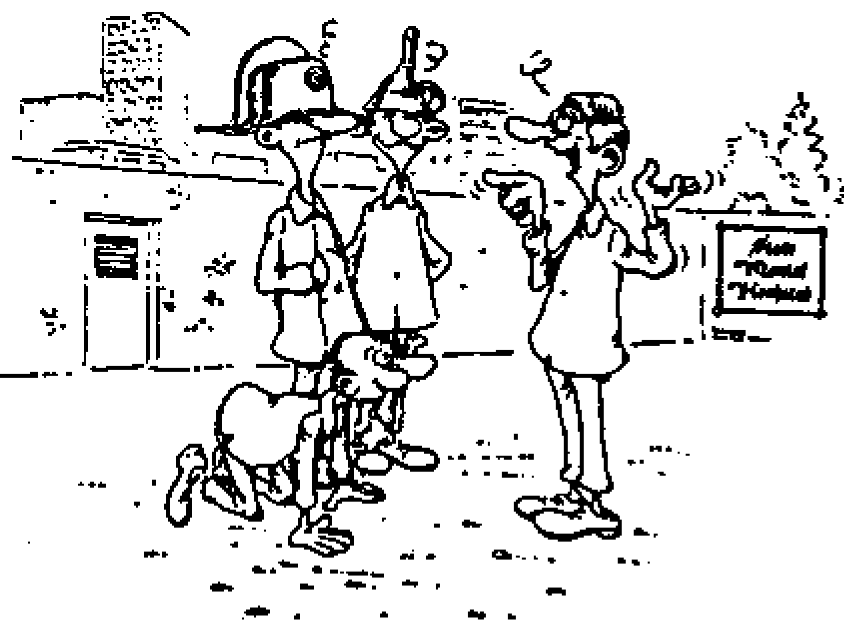
Richard Otto en Nico Stad (regio N-Holland).



Atom versus Electron

Recente dumpacties deden onlangs de computerhandelaren opschrikken. Vele liefhebbers stroomden toe op de aanbiedingen.
Zo ook op de electron waarvan er nogal wat op de markt gebracht zijn
Het is niet zonder reden dat we dit stukje schrijven.
Het is namelijk gebleken dat een aantal Atomgebruikers ook gebruik gemaakt hebben van de aanbieding, en dus een Electron gekocht hebben
Ieder met zijn eigen beweegreden, echter meestal als aanvulling op de Atom.
Gezien de achtergrond en belangstelling van de doorsnee Atomgebruiker (hoe die er dan ook uit mag zien?) blijkt die aanschaf niet zo verwonderlijk.
Het voor ons gewone zinnetje cq begrip hardwarematige aanpassing blijkt in de wandelgangen veelvuldig gefluisterd te worden.(niet iedereen weet van zichzelf dat hij/zij zo'n ding gekocht heeft)
De redactie heeft besloten indien daar belangstelling voor bestaat hardware aanpassingen/koppelingen etc. tussen Atom en Electron te publiceren. Het zal uitdrukkelijk een passief redactioneel gebeuren zijn. We zullen dus niet actief achter dergelijke zaken aangaan!
Stuur dus uw gewaardeerde bijdrage naar de redactie, het adres mag bekend verondersteld worden.
Van onze kant zullen wij ons best doen om een ieder zoveel mogelijk van dienst te zijn.

de redactie



CTRL SHIFT RETURN BREAK INSERT CLEAR GOTO
SELECT START < DELETE BAKS ← SHIFT LOCK
SYSTEM RESET ↑ RETURN LINE FEED ESC RPT
SHIFT ↓ RETURN DS → SELECT INSERT •
ENTER SHIFT ↓ = CTRL START• RETURN • > , 0
CLEAR BACKSPACE TAB PRINT REM . . . .

LOGIKA 3.

Inleiding.

Bespraken we in deel 2 van deze "cursus" de SR-flipflop en zagen we daar een voorbeeld van in de vorm van een dendervrije schakelaar; in dit deel 3 wou ik hier graag nog iets verder op doorgaan. We houden dus nog even vast aan de SR-flipflop, die opgebouwd is met NAND-gates.

Het RAM.

Als tweede toepassing van de SR-flipflop (ook wel SR-latch) zullen we nu een geheugen element bespreken, zoals dat toepassing vindt in een RAM chip. We kunnen ons de schakeling als volgt opgebouwd denken.

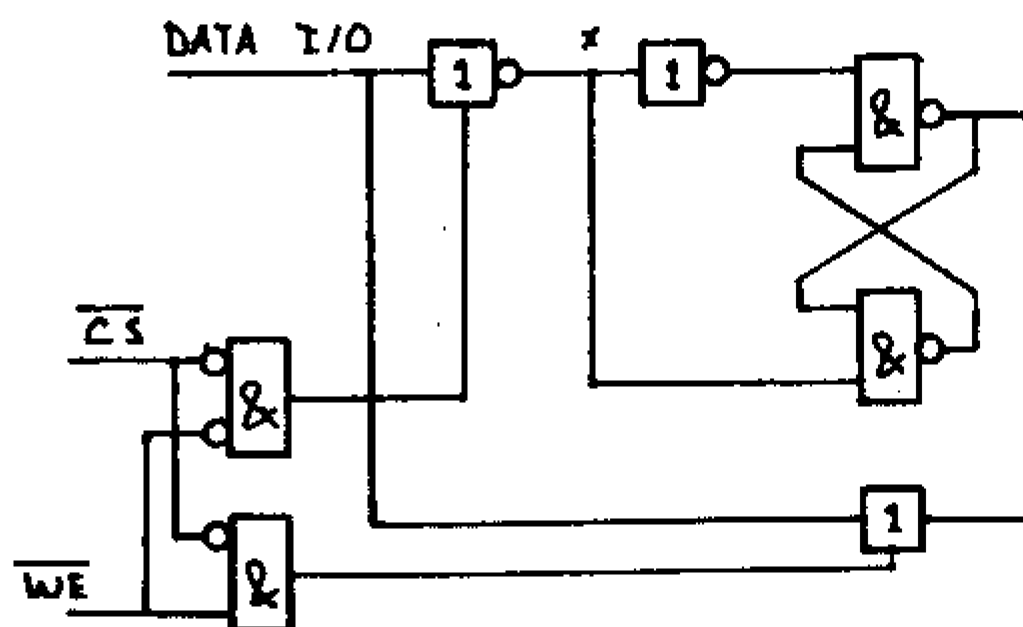


fig.1

Dit schema is als volgt te verklaren.

DATA I/O is het lijntje via welke de DATA (de Ø of de 1) in de flipflop geschreven cq uit de flipflop gelezen wordt.

$\overline{CS}$ is het chip select lijntje. Een chip select lijn is vaak laag aktief. Dwz dat de desbetreffende chip geselekteerd is als Chip Select laag is (vandaar de inversie streep boven CS).

$\overline{WE}$ is het Write Enable lijntje. Ook Write Enable is nor- maal gesproken laag aktief. Als WE laag is, gaat er in de geheugencel geschreven worden.

De AND gates met de inversie cirkels, zijn AND gates met inverterende ingangen. Een AND gate met inverterende ingangen, welke laag zijn, gedraagt zich

net zo als een AND gate met gewone ingangen, welke hoog zijn.
De bouwsteen rechtsonder is een buffer met 3-state uitgang. Dit is een uitgang, welke normaal werkt, als de derde aansluiting (de onderste) laag is. Als deze ingang (control) hoog is, dan is de bouwsteen non-aktief / uitgeschakeld (de uitgang is dan hoogohmig). Zo kunnen er bouwstenen parallel geschakeld worden, zonder dat er kans op kortsluiting bestaat. Misschien ten overvloede: een buffer is een bouwsteen, welke de informatie onveranderd doorgeeft. De uitgang is dus hoog als de ingang hoog is.

Het hart van de schakeling is de SR flipflop met de inverter erbij. Als het punt gemerkt x hoog is, dan is Q ook hoog. Als dit punt laag is, dan is Q ook laag (zie funktietabel SR flipflop).
Door nu CS laag te maken, wordt de geheugencel geselekteerd. Afhankelijk van de toestand van WE, gaat er gelezen (READ) of geschreven (WRITE) worden. Stel, WE is ook laag. Control van de 3-state buffer in de DATA I/O lijn is nu laag. De buffer is dus aktief. De informatie van DATA I/O wordt nu doorgegeven aan de SR latch. Als dit "schrijven" heeft plaatsgevonden, kan CS hoog gemaakt worden. De toestand van WE doet er nu niet toe. De geheugencel is niet meer geselekteerd. De informatie van DATA I/O zit nu in de SR latch en blijft er in zitten, totdat er opnieuw geschreven gaat worden (of totdat de buurman zijn 10 jaar oude Philips KTV inschakelt, waardoor in het hele flatgebouw het licht even dimt en onze komputer even zijn spanning kwijtraakt ...).
Nu willen we na verloop van tijd wel weer eens weten, wat we tevoren in de geheugencel geschreven hebben. Hiertoe maken we CS weer laag, zodat de geheugen cel weer geselekteerd is. Voordat we CS laag maken, moeten we echter zorgen, dat WE hoog is. Anders wordt er voordat we gaan lezen, eerst nog nieuwe DATA geschreven (WE laag). Dit is natuurlijk niet gewenst. De cel staat nu in de READ stand en de informatie (DATA) kan nu van de DATA I/O lijn afgehaald worden. Bij deze aktie blijft de toestand van de flipflop onveranderd.

Nu is een geheugen celletje natuurlijk niet spektakulair, maar gelukkig laten geheugencelletjes zich bundelen en dankzij de 3-state logika, laten geheugen celletjes zich parallel schakelen. Nu is bijv. meneer INTEL zo vriendelijk dit voor ons te doen en dit geheel in een huisje onder te brengen. Zo ontstaat dan bijv. een, voor velen van ons bekende, chip die de naam 2114 draagt. De gebundelde geheugen celletjes worden dan een memory array genoemd (zie figuur 4).
Om nu bepaalde geheugencelletjes uit te kiezen (bij de 2114 worden er steeds 4 tegelijk gekozen), vindt er via de adreslijnen een selektie plaats. Zo kunnen er 1024

kwartetten geheugen cellen gekozen worden.
Door nu 2 2114 RAM chips naast elkaar te zetten, krijgen we 1024 achttallen (bytes) geheugen cellen. Zo ontstaat 1 kByte (kilo) statische RAM. Statisch betekend, dat er niets extra's gedaan hoeft te worden, om de geheugencel zijn waarde te laten behouden (de SR flipflop houdt zijn waarde zelf vast, zoals we gezien hebben).
RAM betekent Random Acces Memory, oftewel willekeurig toegankelijk geheugen, oftewel lees/schrijf geheugen.
Tenslotte als toetje de pin konfiguratie (fig.2), het logische symbool (fig.3) en het blok diagram (fig.4) van onze "geliefde" 2114. Noem het een info kaart in een info blad.

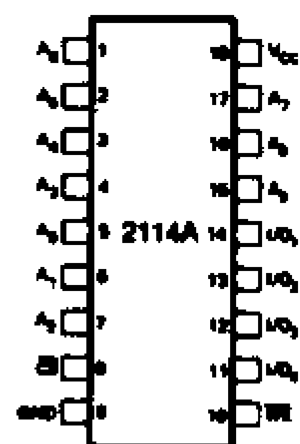


FIG. 2

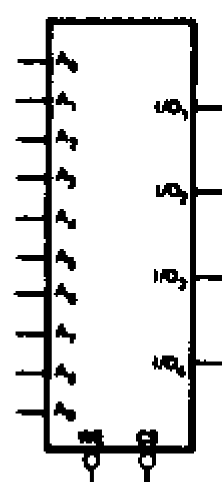


FIG. 3

PIN NAMES

A_0–A_9	ADDRESS INPUTS	V_{CC} POWER (+5V)
$\overline{WE}$	WRITE ENABLE	GND GROUND
$\overline{CS}$	CHIP SELECT	
I/O_1–I/O_4	DATA INPUT/OUTPUT	

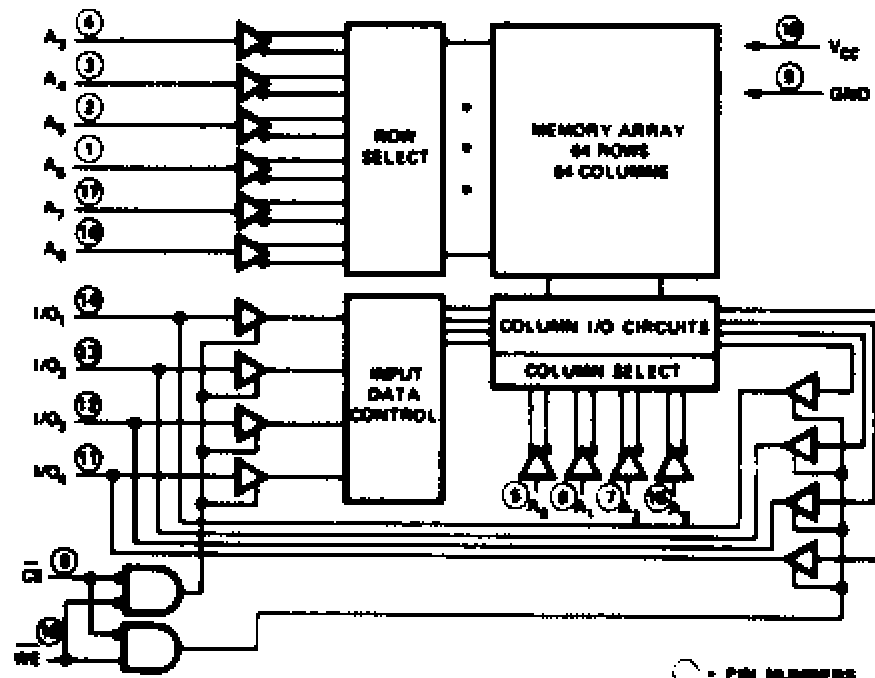


FIG. 4

Samenvatting.

Met een SR flipflop kunnen we op eenvoudige wijze een geheugencel voor lees/schrijf gebruik maken. De werking ervan is eenvoudig te begrijpen, indien de werking van een SR flipflop (het hart van de schakeling) bekend is. Geheugen cellen kunnen gemakkelijk in 1 IC samengenomen worden. Zo ontstaat een RAM chip. En RAMmetjes zijn nog steeds nodig voor tikgrage vogels, die hun geestelijke kreativiteit in een leuk programma kwijt willen, om dit vervolgens in het periodiek van de Acorn Computerclub Overijssel/Gelderland te publiceren.

Wat mij betreft, tot de volgende keer. Logische groeten,

Wibo Visser.

65816 Microprocessor

```
 ------------------------------------------------------------------
 :                                                                :
 : toekomstdromen: -Atom als 8/16 bits machine                    :
 :                 -Atom met 16 Mbyte adresseerbereik             :
 :                 -Atom en Multiprocessing CP/M etc.             :
 :                 -Atom en een goedkoper geheugen                :
 ------------------------------------------------------------------
```

In deze en volgende artikelen wil ik graag iets vertellen over de nieuwe microprocessor van Western Design Center, nl. de W65SC816, welke ontworpen is om verouderde (???) 6502-computers nieuw leven in te blazen.
Het ontwerp is er op gebaseerd dat deze pin- en software (upwards) compatible is met de 6502, zodat beschikbare software bruikbaar blijft.
Wanneer men de vakliteratuur volgt, dan valt op dat de prijzen van geheugenchips kelderen, dat 64 Kbyte werkgeheugen niets bijzonders meer is en dat Megabyte systemen reeds hun toepassing vinden in de PC systemen.
De processoren 6502, 6800 en Z80, welke enkele jaren terug nog voor "State of the Art" systemen doorgingen, zijn heden ten dage sterk verouderd.
Daaruit volgt echter niet dat we de Atom maar bij het grootvuil moeten zetten want er zijn in het sterk uitgedunde kamp van computerfabrikanten een aantal belanghebbenden voor een pin- en software compatible processor voor de 65(C)02, welke 16 bits kan `draaien´ met een zeer groot adresseerbaar geheugen zonder dat Bankswitching moet worden toegepast.

Belanghebbende giganten zijn oa. Apple en Atari, waarvan de eerstgenoemde een enorme reputatie heeft opgebouwd in de fabrieken en scholen als een veelzijdig multi-systeem.
Wanneer hiervoor een 8/16 bits compatible processor ontwikkeld zou worden welke zonder al te veel aanpassingen kan functioneren, dan liggen de gevolgen voor de hand....
Western Design lijkt hierin geslaagd te zijn, ze hebben evenals destijds bij de ontwikkeling van de 65C02 nu weer een pin-compatible processor ontwikkeld maap met een ander doel, nl. als concurent voor de 8086 en de 68000 processoren.
De vraag welke direkt ontstaat is: hoe is het mogelijk om een 8-bits computersysteem om te bouwen tot een 16-bits machine?
Hierover later meer, maar je kunt het vergelijken met schakelen van buslijnen, waarbij op het ene moment het MS-byte en op het andere moment het LS-byte op de bus wordt gezet.
Wellicht zullen het programmeermodel samen met de pin-configuratie enige duidelijkheid bieden.
Het streeplijndeel van de tabel is datgene wat de W65SC816 (in het vervolg de 65816) meer te bieden heeft dan de 6502. Het status register heeft er een vlag bij gekregen, nl. de E of Emulation vlag, welke zorgt voor omschakeling van 6502 - 65816 mode.

Deze omschakelmogelijkheid dient ervoor om van de voordelen van beide systemen (8/16 bits) te kunnen profiteren en ... om oude software te kunnen gebruiken welke anders rijp voor de prullemand zou zijn. Wanneer er gesproken wordt over voordelen, dan heeft dit in dit geval ook betrekking op snelheid.
Deze processor is in staat om met klokfrequenties te werken in het gebied van 1 ... 10 MHz!
Ook dit is een belangrijke verbetering tov. de 6502.

Overzicht tav. het programmeermodel:

Fig.1 laat ons het programmermodel zien van de 65816. Het gaat hier om een uitbreiding van de 6502 registers, nl.:
*De 8-bits 6502 registers zijn 16-bits geworden;
*De 16-bits Programcounter is nu 24-bits (geen 32-bits zoals de 68000 ea.);
*De niet-gebruikte bits in het statusregister zijn gevuld, waarbij het E-bit is toegevoegd;
*Behalve de Accu's zijn de originele 6502 aanduidingen X, Y, S, PC en D gelijk gebleven, echter met H en L toevoegingen voor de High- en Low-order bytes:
De complete accu heet nu C, met B als 'H' en A als 'L' byte.

1. Extra registers.

Een extra 16 Bits register (Direct register) duid de plaats aan van de DIRECT PAGE (ook wel zero page genoemd). Dit is een 256 byte groot gebied waarin de snelst uit te voeren bewerkingen kunnen worden opgenomen, immers het adres in deze pagina wordt in een enkele byte gedefinieerd.
Het aantrekkelijke van de 65816 is dat de Direct pagina op elke gewenste plaats in het geheugen gezet kan worden!!! Bij de 6502 zit die pagina ALTIJD helemaal onderin het geheugen (van #0000 t/m #00FF).
Ook nieuw zijn de twee 8-bit Data Bank Registers (DBR) welke het gebruik van het volledige 16 MByte geheugenbereik mogelijk maakt.
De programcounter is uitgebreid met 8-bit, d.m.v. het PBR-register en vergroot zo het (sprong) bereik van deze counter.

2. Instructieset.

In de instructieset van fig.1 zijn alle instructies van de 65816 weergegeven waarbij typisch-65816 instructies met een ster (*) en 65C02 instructies met een stip (.) zijn aangegeven; totaal 265 stuks. De standaard 6502 instructies hebben geen toevoeging.
In de tabel staat voor elke opcode de Hex-waarde, de Mnemonische aanduiding en de adresseringsmode. Het aantal klokperioden benodigd voor uitvoering van de instructie staat rechts onder in elk hokje en het aantal bytes dat elke opcode + operand nodig heeft staat links daarvan.

3. Nieuwe Instructies.

*** Blockmove instructies ***

*** MVN MVP ***

Om direkt maar van wal te steken met de MVP (MoVe Preceeding) en de MVN (MoVe Next) instructies, welke Z-80 achtige instructies zijn (!!) hoop ik ook de mensen tevreden te stellen die tot nu toe van dit artikel met enige reserve kennis hebben genomen.
MVP verplaatst een datablok hoger in 't geheugen, waarbij de verplaatsing start aan de "top" van het blok en gaat vervolgens verder naar beneden.
Bij MVN is de startpositie onderaan het blok en gaat verder naar boven, verder zijn MVP en MVN identiek.

*** Onvoorwaardelijke Branch ***

*** BRA ***
De gewone 6502 mist een BRA (Branch Relative Always). Vergeleken met een JMP (jump) instructie is er een fundamenteel verschil, n.l. dat een Branch RELATIEF is, d.w.z. de sprong wordt gerelateerd aan de stand van de program counter en niet zoals bij een JMP verbonden aan een vast adres.

Het voordeel van een relatieve branch is dat bij het reloceren van het programma (= omzetten naar een ander deel van het geheugen) er geen nieuwe adressen behoeven te worden berekend. Een nadeel van een relatieve branch is dat het sprongbereik slechts 265 geheugenplaatsen is, resp. 129 plaatsen vooruit en 126 plaatsen achteruit.

*** Datatransport tussen de interne registers ***

*** PHX PHY PLX PLY TXY TYX ***
De PHX en PHY (PusH index X-Y to stack) en PLX - PLY (PulL index X-Y from stack) instructies vereenvoudigen het uitwisselen van data tussen stack en index-registers.
TXY (Transfer X to Y) en TYX (Transfer Y to X) verzorgen de datatransport tussen de index-registers.

*** Bitmanipulatie ***

*** TRB TSB REP SEP ***
De instructies TRB (Test and Reset Bits) en TSB (Test and Set Bits) worden gebruikt om op afzonderlijke bits te testen of deze te veranderen.
De REP en SEP instructies zijn gelijksoortig, echter dan voor het conditiecode (status) register.

*** Stackpointer gebruik; data van/naar de stack ***

*** PHB PLB PHD PLD PHK * PEA PEI PER ***
Deze 8 nieuwe instructies maken een eenvoudig stackgebruik mogelijk; 5 van de 8 instructies verzorgen transport van en naar de stack; de andere 3 maken het mogelijk om verwerkte data "even" op de stack te zetten en daarna terug te zetten in een der processor registers.
PHB (Push DBR on stack)   DBR = Data Bank Register
PLB (Pull DBR from stack)
PLD (Pull Direct register from stack)
PHK (Push program bank register from stack)
PHD (Push Direct register on stack)
De andere 3 instructies zijn de volgende:

PEA (Push Effective indirect Adress) welke resp. de 2e en 3e byte van de instructie in de Direct Page, een offset die naar de 1e van 4 opeenvolgende bytes wijst welke op de stack gezet moeten worden.
PER (Push Effective programcounter Relative adress) dient er voor om het adres van een datablok door te geven aan een subroutine. De data wordt daarbij opgeslagen in de oproepende routine en het adres wordt als een offset t.o.v. de PER instructie doorgegeven.

*** TCS TSC ***
De 65816 z'n TCS (Transfer accu C to Stack) en TSC (Transfer S to C) instructies maken een eind aan het euvel van de 6502, waarbij de stack alleen via het X-register bereikbaar is.

*** Instructies voor 16 MByte groot adresseer bereik ***

*** TCD TDC ***
Omdat de 65816 meer interne registers heeft dan de 6502 zijn er nieuwe instructies nodig voor intern datatransport. De TCD (Transfer accu C to Direct register) en de TDC (Transfer D to C) instructies bieden de enige mogelijkheid om het direct register te lezen zonder stackgebruik.

*** JSL RSL ***
Omdat het adresseerbereik van de 65816 groter is dan die van de 6502 moet ook de subroutine "sprong" vergroot kunnen worden (>65536 plaatsen). Hiervoor dienen de instructies JSL (Jump Subroutine Long) en RSL (Return from Subroutinr Long).

*** XBA XCE ***
De Instructies XBA (eXchange B and A) verwisselt de MSB en LSB van de accu. XCE (eXchange C and E flag) verwisselt de Carry (C) met de Emulation (E) vlag in het conditie code register.

Resumerend kan nu reeds opgemerkt worden dat de besproken extra's van deze processor zeker de moeite waard zijn, ook al gebruikt men niet alle mogelijkheden zoals b.v. het grotere adresseerbereik omdat momenteel geheugenchips nogal prijzig zijn.

De volgende en laatste aflevering over deze processor wordt bespoken:
- nieuwe interrupt routines;
- adresseer modes;
- hardware veranderingen;
- multiprocessing.

Jaap v/d Veen

****

C A S T L E   Q U E S T

****

Naar aanleiding van het artikel ADVENTURES van G. Hillebrand heb ik mijn gedachten laten gaan over het ontwerpen van een GRAPHIC ADVENTURE.
Aangezien ik niet in het bezit ben van een diskdrive verviel de mogelijkheid om graphische voorstellingen van de omgeving van de schijf in het beeld te laden.

Om deze beelden in het geheugen van de ATOM te zetten leek me ook geen goed idee want om een beetje gedetailleerd landschap op te slaan ben je al gauw 2 K per tekening kwijt.
Wanneer je dus een ATOM hebt met alleen laag en hoog gestapeld geheugen, kun je ongeveer 4 a 5 schermen opslaan omdat de rest van het geheugen voor het programma beschikbaar moet zijn en in een ADVENTURE met 5 schermen ben je ook zo "uitgeadventured"

Nee, ik moest dus iets anders verzinnen.
Toen zag ik Castle Quest op de BBC, en de opbouw van deze ADVENTURE leek me wel wat.
De opbouw is als volgt:
Je bent in een kasteel met trappen, muren, vloeren, monsters en nog veel en veel meer.
Wanneer je door het kasteel loopt scrolled het scherm mee.
Het lijkt dus of je je in een ontzettend groot kasteel bevind, terwijl er maar relatief weinig schermen in het geheugen hoeven te zitten, de schermen worden dus aan elkaar gekoppeld d.m.v. een scrollroutine.
Deze opbouw heb ik ook ge- (mis) bruikt voor een Castle Quest versie voor de ATOM.
Het hele kasteel is opgebouwd uit 9 verschillende schermen die m.b.v. 4 scrollroutines, die jammer genoeg veel te traag zijn om hetzelfde effect te verkrijgen als bij de BBC, aan elkaar gekoppeld worden.

Het resultaat is een kasteel met 63 velden geworden wat vrij eenvoudig uit te breiden is.
Dit uitbreiden is vrij eenvoudig omdat in het programma 2 ARRAY's zitten, nl. een ARRAY waarin het beginadres van het scherm zit en een ARRAY waarin de situatie bijgehouden wordt, bv. in scherm 4 ligt een goudstaaf of in scherm 32 bevindt zich een monster.
Net voordat het nieuwe scherm in beeld gescrolled moet worden, wordt de situatie al in het nieuwe scherm gezet en nadat het nieuwe scherm in beeld is wordt de situatie in het scherm in het geheugen weer gewist.
In het programma bevindt zich een loop om het figuurtje in het beeld te laten bewegen, waar meteen getest wordt wat er met de aanwezige situatie gedaan moet worden.

Castle Quest is een eerste aanzet tot een GRAPHIC ADVENTURE.
Ik hoop dat je er veel ADVENTURE plezier mee beleeft.

Tot slot wil ik nog enkele spelregels verraden:

-De bedoeling is om uit het kasteel te ontsnappen.
Dit kan doordat men met de SLEUTEL bij het SLOT komt.

-De SLEUTEL bevindt zich ergens in het kasteel en kan pas dan opgepakt worden wanneer men alle 12 de goudstaven gevonden heeft.

-De HONDEN zijn te verslaan door een ZWAARD wat ergens in het kasteel ligt op te halen.

-De VOGELS kan men alleen ontwijken.

-Wanneer de punten op zijn komt het SLOTSPOOK u halen.

Het spel maakt gebruik van de volgende geheugengebieden:

hoofdprogramma	#2900 - #3B86
niet gebruikt	#3B87 - #3C11
machinetaal routines	#3C12 - #3FFF
beeldscherm	#8000 - #85FF
opslag schermen	#8600 - #97FF
niet gebruikt	#9800 - #98FF
sprites	#9900 - #9A3F
array's	#9A40 - #9C3F
niet gebruikt	#9C40 - #9CFF
tekstscherm	#9D00 - #9EFF
niet gebruikt	#9F00 - #9FFF

Wanneer men geen schakelkaartsysteem heeft moet het spel als volgt opgestart worden:

```
*LOAD"CASTLE QUEST"
CLEAR2;*LOAD"DATA"
GOTO 230
```

Wanneer men wel een schakelkaartsysteem heeft:

```
*RUN"CASTLE QUEST"
```

en het programma laadt automatisch de DATA binnen.

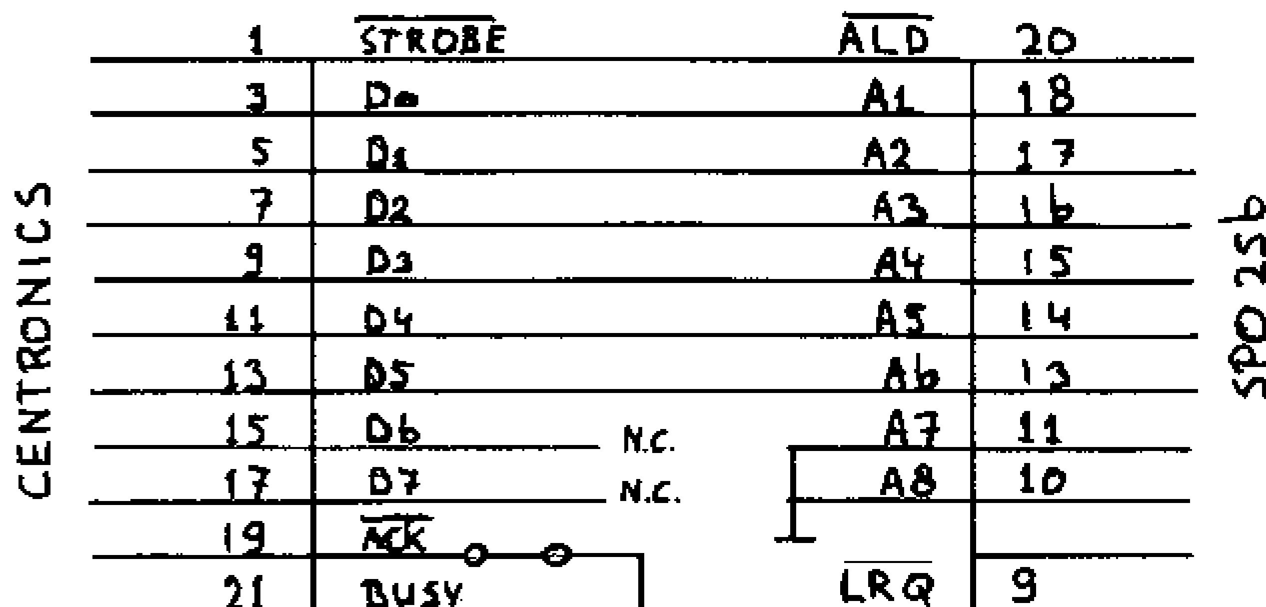
Indien er vragen en/of opmerkingen zijn over Castle Quest bel dan even naar Kees van Oss , tel. 04104-74323

V E E L P L E Z I E R

PROJEKT:SP0256-AL2 op Centronics poort Bram Poot

Als u Acorntjesbrood en de andere regiobladen goed gelezen heeft, weet u dat er al een heleboel koppelingen aan de Atom van het spraak-IC SP0256-AL2 zijn beschreven.
Waarom dan nog een keer deze materie behandeld?
Omdat de hier te beschrijven methode geen (of hooguit nauwelijks) extra software nodig heeft. Er wordt namelijk gebruik gemaakt van de Atom printerpoort.
Voordelen:
1. software reeds aanwezig
2. gebufferde signalen (verbinding mag meters lang zijn)
3. universele koppeling (ook op andere computers te gebruiken)
Nadelen:
1. VIA (IC1), 74LS244 (IC50), printerconnector en printerkabel nodig
2. printer en spraak-IC niet tegelijkertijd te gebruiken
3. niet "rack-compatible" (wat dat ook moge betekenen)

Voor ons (een klein groepje TH-studenten) golden en gelden de nadelen niet, zodat de keuze van implementatie wel duidelijk is.
Het principeschema ziet er aldus uit:



Hoe de rest van het spraak-IC moet of kan worden aangesloten, kunt u in de eerder vermelde bronnen lezen.
De software schittert door eenvoud:

```
10 REM SPREEK!
20 RESTORE
30 PRINT $2
40 DO READ X
50 PRINT $X:#80
60 UNTIL X=#80
70 PRINT $3
80 END
90 DATA 20,2,8,58,11,#80
```

De printerdriver zorgt voor de handshake (die overigens de VIA enigszins tekort doet), zodat u zich geen zorgen hoeft te maken over de tijdsduur van de fonemen. Pas als het IC uitgekletst is, wordt de volgende klank verstuurd.

Het kan natuurlijk wel wat fraaier:

```
 10 PROGRAM SPREEK!
 20 PROC SPEAK(N),I
 30   FOR I=1 TO N
 40     A=FF(I) OR #80
 50     LINK #FEFB
 60   NEXT I
 70 PEND
 80 !#208=!#208+3
 90 READ X
100 DIM FF(X)
110 FOR I=1 TO X
120   READ FF(I)
130 NEXT I
140 PRINT $2
150 SPEAK(X)
160 PRINT $3
170 !#208=!#208-3
180 DATA .. ; REM AANTAL FONEMEN
190 DATA ..,..,..,..,..
200 DATA ..,..,..,..,.. etc.
```

INTIKKEN EN RUN

DOP.#C.#(C.>31);U.C.=0

Als u tijdens het uitspreken van een foneem de processor i.p.v. een eeuwigheid te laten wachten, nog iets nuttigs wilt laten doen, dient u de routine #FEFB te herschrijven tot iets soortgelijks, maar dan zonder BUSY-check.
Als u de ACKNOWLEDGE en BUSY verknoopt, kunt u zelfs onder interrupt spreken. Ik vermoed echter dat het computerge-OH dermate snel gaat vervelen dat u hier geen behoefte aan zult hebben.

Het bovenstaande verhaal is geschreven nog voordat er ook maar 1 SP0256 daadwerkelijk aangesloten was. Toen dat eenmaal wel het geval was, is er een ware wildgroei ontstaan inzake de spraaksoftware. Een van de gemeenschappelijke onderdelen van de diverse benaderingen betrof het spreken onder interrupt, waarbij de klanken in een buffer worden bijgehouden. Het programmeren van deze zgn. queue is tamelijk ingewikkeld (kwalitatief niet, maar in concreto wel degelijk), zodat het voordeel van de eerder vermelde eenvoudige software inmiddels schijnbaar als een lachertje moet worden gezien.
Schijnbaar, omdat er zeer zeker toepassingen zijn waarbij het reeds aanwezig zijn van software bijna als een noodzaak moet worden beschouwd.
De voordelen van de queue moeten daarentegen ook weer niet onderschat worden. De huidige oplossing maakt het mogelijk om tijdens een programma (bijna) zonder tijdverlies hoorbare boodschappen aan de programmeur danwel spelletjesspeler te geven. Denk in dit verband bv. aan een "adventure" waarin de beschrijving van de ruimte hoorbaar wordt gemaakt, terwijl de computer intussen de nodige zaken instelt, controleert en/of aanpast.
Elders in dit blaadje staat een programma dat heel netjes zo'n queue aanmaakt en afhandelt. Het werkt zelfs samen met andere interrupts en ik moet zeggen dat het een waar genoegen is om de VIA in een dergelijke situatie zo druk bezig te zien.

SPRITES OP DE ATOM.

Velen van u zullen inmiddels wel weten wat sprites zijn en hoe gemakkelijk er animaties op het beeldscherm mee zijn te programmeren. M.b.v. Gags heeft u immers de nodige routine inzake sprite-handling kunnen opdoen. De sprites van Gags zijn weliswaar erg mooi, maar het zijn toch geen echte sprites en dat komt omdat e.e.a. in software moet plaatsvinden vanwege een voor dit doel ongeschikte hardware-configuratie van de Atom.
Maar daar kunnen we wat aan doen en wel via de VDU-chip TMS9929 van Texas Instruments. Dit IC biedt een aantal aardige faciliteiten:
-> grafics (256*192)
-> tekst (40* 24)
-> sprites (32 stuks)
-> programmeerbare karakters
-> kleur (15 kleuren + doorzichtig)
-> 16k "transparant" geheugen
-> eenvoudige implementatie

Het IC verlangt twee bytes in de memory map, via welke tevens het VDU-geheugen wordt beschreven c.q. gelezen. Een leuke bijkomstigheid van dit IC is bovendien dat het als MSX-standaard is gebombardeerd, hetgeen dus inhoudt dat we hiermee in principe MSX-software kunnen draaien. De hardware is daarbij van ondergeschikt belang, als we maar de beschikking hebben over een analoge implementatie van de MSX-VDU-statements, zoals SPRITE, CIRCLE, PAINT en COLOR. En als we eenmaal de beschikking hebben over de kaart (eenvoudig op te bouwen) en de onderhavige statements (iets moeilijker te implementeren) dan heeft het weer zin om tijdschriften te kopen, aangezien daar een overvloedige hoeveelheid MSX-software in staat, die wij dan tot op zekere hoogte eveneens kunnen draaien.
In hetzelfde kader valt overigens te streven naar een muziekkaart rond de AY-3-8910, ook een MSX-toeter c.q. -bel.

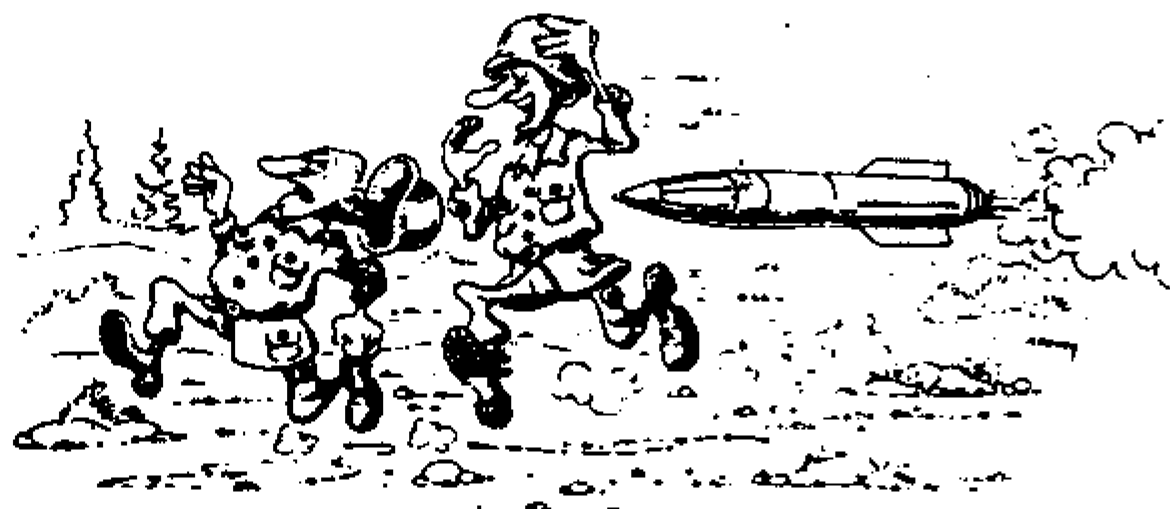
Laten we nu iets dieper ingaan op het bewuste IC.
In BYTE (augustus 1982) heeft een artikel + schema + software gestaan rond de TMS9918, waaruit dit project is ontstaan. Bij bestelling van de chip bleek dat de 9929 "beter" was, omdat deze een PAL-uitgang heeft en de 9918 een NTSC (Never The Same Color)-uitgang.
Een groot voordeel van deze IC's is dat ze RAS- en CAS-signalen leveren, zodat een dynamisch geheugen eenvoudig toegepast kan worden, waarmee de kaart redelijk betaalbaar blijft. Een nadeel van dynamisch geheugen is dat er voedingen van +12 en -5 Volt nodig zijn. Voor deze laatste kan het "spiegel-IC" ICL7660 aangewend worden; +12 Volt komt uit de floppy-interface of uit de RS232 of moet natuurlijk zelf bijgebouwd worden.

T.a.v. de sprites en hun opbouw zijn een paar dingen 't vermelden waard. Ze bestaan uit 8*8 of 16*16 pixels, instelbaar op twee grootten. Elk van de 32 sprites heeft een eigen prioriteit, zodat je ze over elkaar heen kunt laten schuiven. Als de ene sprite een andere raakt (dat gebeurt als ze minimaal 1 pixel overlappen), kan dit via een interrupt kenbaar gemaakt worden.

De sprite-handling wordt volledig verzorgd door de 9929; het enige dat je zelf moet doen is 1 keer de vorm definieren, waarna de kleur, plaats en beweging eenvoudig in de 9929-registers gePOKEd kunnen worden.

De grafische mode wordt opgedeeld in 32*24 blokjes van 8*8 pixels. In elk blokje kan via een attributentabel een willekeurige figuur geplot worden,dus ook karaktervormen als je dat wilt.
Aangezien de tabel 768 patronen kan bevatten,is het beeldscherm volledig bestuurbaar. In feite worden tekst, grafics en sprites in deze mode op onderling gelijksoortige manier opgebouwd,zodat ze op het scherm "gemengd" kunnen worden.
Overigens is deze grafische mode nog opgedeeld in I en II. Mode I laat een tabel toe tot 256 patronen en 2 kleuren per blokje.
Mode II geldt voor de 768 patronen en staat 2 kleuren toe per byte, dus maximaal 16 kleuren per blokje.
In de tekstmode wordt het scherm opgedeeld in 40*24 blokjes van 6*8 pixels. De attributentabel kan nu echter maar 256 patronen bevatten. Aangezien er 960 blokjes op het scherm passen is niet elk denkbaar beeld op te bouwen. Bovendien kunnen in deze mode geen sprites ingezet worden.
Verder bestaat er nog een low-resolution mode van 64*48 blokjes (4*4 pixels) die per stuk een van de 16 kleuren kan hebben.
Het mag duidelijk zijn dat de grafische mode II het meeste geheugen nodig heeft en wel ongeveer 12k. Aangezien er 16k aanwezig is, gebruikt of niet gebruikt, is dat geen probleem. Echter, een toepassing die maar twee kleuren gebruikt met 2*6 patronen van 8*8 pixels en 32 sprites heeft nog geen 4k nodig. Door het simpelweg veranderen van pointers in de registers van 9929 kan geschakeld worden tussen 4 beeldschermen, een toegevoegde faciliteit.

Op dit moment bestaat er een werkende sprite-kaart die zelfs al is gedemonstreerd op een regioavond. De BYTE-software kon nagenoeg zonder aanpassingen worden ingetikt en m.b.v. de symbolische assembler worden vertaald.
Over het IC en de software (ook de (nog) niet bestaande) kunnen nog pagina's vol geschreven worden. Als inleiding wilden we het hier voorlopig even bij laten. Bij gebleken belangstelling vertellen we misschien ooit nog wel het een en eventueel ook nog
het ander.



Just a little bug in the guidance program, Sarge!

Deze keer alweer de derde aflevering uit de serie van x over de grootste gemene deler van de Atoms van ieder van ons. We willen deze keer naast "standaard" hardware ook enkele software-aspecten bespreken.
Wat we nog niet met zoveel woorden hebben besproken, zijn de "standaards" die allang bestaan, t.w. de acht-bits-printerinterface en de joystick-aansluiting. Deze beide vallen in de als-dan-categorie.
Dus als u een 8-bits-printerinterface maakt dan wordt u geacht dit te doen via PC3 van de 8255 en als u een joystick-aansluiting maakt dan wordt u geacht dit via PB0...PB4 van de 8255 te doen. Het hèbben van deze aansluitingen wordt (nog) niet voorgeschreven. De aansluitgegevens van beide uitbreidingen worden na dit artikel nogmaals separaat gepubliceerd, waarbij we dankzeggen aan de auteurs ervan. De ervaring heeft geleerd dat, omdat deze aansluitingen ooit "gestandaardiseerd" zijn, er geen aanpassingsproblemen zijn gerezen bij de erop toegepaste software.
Ook i.v.m. de te verwachten hardware-uitbreidingen moeten we de vinger aan de pols houden inzake programmeertechnieken.
Zoals inmiddels genoegzaam bekend, heeft Acorn de Atom zodanig ontworpen dat hij uitbreidbaar is. Eén van de methoden die daarvoor gebruikt zijn, is het concept van de gevectoriseerde systeemroutines.
Het is van het grootste belang dat we, waar mogelijk, dit concept op dié manier gebruiken waarvoor het bedoeld is. Doen we dit niet dan zullen we vrijwel elk programma aan veranderende configuraties moeten aanpassen, terwijl dit, als we het "goed" doen, lang niet altijd nodig is.
Een voorbeeld.
Vele machinetaalprogramma's roepen de routine ≠FE52 aan als ze een karakter op het scherm moeten plaatsen. Dit gaat alleen goed als op dat moment het orginele Atom-beeldscherm gebruikt wordt. Als er echter een andere beeldschermcombinatie is ingeschakeld, gaat er vrijwel altijd van alles mis.
Het is daarom veel en veel beter om de officiële write-character routine op ≠FFF4 aan te roepen. Op deze manier werkt het programma onafhankelijk van welk beeldscherm wordt gebruikt en hoeft dus niet aangepast te worden aan b.v een 80-kolommenkaart of GRMOD. Dit alles natuurlijk op voorwaarde dat de systeemroutines op de "officiële" manier zijn geïmplementeerd.
Wat dit aangaat slaat b.v. P-charme enigszins de plank mis door in INKEY de routine op ≠FE94 aan te roepen. Hierdoor komt iemand die b.v. een ASCII-toetsenbord aan zijn (of haar) Atom heeft aangesloten en de read-character-routine dus heeft herschreven in de problemen en moet P-charme aanpassen. Dit zou niet gehoeven hebben als de officiële read-character-routine ≠FFE3 werd aangeroepen.
Gezien de kwaliteit van P-charme zal de auteur wel z'n redenen gehad hebben om het zó te doen, maar toch vinden wij dat, waar mogelijk, de Acorn-lijn gevolgd dient te worden.
Een soortgelijk geval (maar toch ook weer anders) doet zich voor als een programma ≠FD69 aanroept om het scherm schoon te maken. De officiële weg is: LDA @12; JSR ≠FFF4 in machinetaal of P.$12 in basic. Deze methode is I/O-onafhankelijk en daarom te prefereren.
Er zijn vele voorbeelden te verzinnen (en uit de praktijk te halen) die laten zien welke (vaak onnodige) problemen kunnen worden verwacht.
Misschien dat de praktijkvoorbeelden af en toe bekritiseerd kunnen worden (in de gunstige betekenis van het woord). Probeer in ieder geval zoveel mogelijk de software zo min mogelijk afhankelijk te maken van de computerconfiguratie.

Voor systeemsoftware houdt dit in dat deze "gedefinieerd" moet zijn, d.w.z. dat op bekende lokaties bekende akties worden ondernomen. Voor toepassingssoftware houdt dit in dat deze gebruik maakt van deze bekende lokaties.
De betekenis van de vorige alinea zal voor velen misschien niet helemaal duidelijk zijn. Als u in basic programmeert heeft u er echter niets mee te maken, want u handelt op dat moment per definitie volgens de regels. De machinetaal-programmeurs zullen daarentegen maar al te goed weten wat we bedoelen en hoeven alleen nog de discipline op te brengen om hun programma's kritisch te bekijken en er niet te gauw tevreden over zijn als het "alleen maar werkt".
Denk niet dat deze opmerkingen nogal vergezocht zijn. Als iemand straks de club-80-kolommenkaart gaat aansluiten zal hij b.v. merken dat:

- de cassette-indicaties van Josbox en P-charme niet meer (naar behoren) functioneren
- het programma COMPACT (diskgebruikers) geen drivenummer afdrukt
- nogal wat programma's op het nieuwe scherm ineens niet meer blijken te werken, terwijl dat vaak wel mogelijk geweest zou zijn. (Een mooi voorbeeld is de niet-grafische BACKGAMMON).

Tot op de dag van vandaag konden we uitgaan van bepaalde Atom-aspecten die bij iedereen dezelfde waren. In de nabije toekomst zal dat steeds minder het geval zijn; denk maar aan nieuwe beeldschermcombinaties, double density disk interface en de talloze cassetteformaten.
Het is zaak om de deelverzameling van ons allen zo groot mogelijk en in stand te houden.
Hopende op een vruchtvolle medewerking uwerzijds etc. etc.,

Bram Poot

P.S. Oorspronkelijk werd bovenstaand verhaal namens de hardware commissie geschreven. Aangezien de commissie het echter goed- noch afgekeurd heeft, hebben de redactie en ik in overleg besloten om het weliswaar te plaatsen, maar van mijn kant uitdrukkelijk onder eigen naam.

STATISTIEK;LOKATIE- en SPREIDINGSMAATSTAVEN.

Dit programma draait op een standaard ATOM met een Floating Point-ROM en heeft ongeveer 3,5 tot 4K aan geheugen nodig, een beetje afhankelijk van het aantal gegevens dat het programma moet verwerken.
Heb je geen Floating Point dan kun je bij alle variabelen waarbij een %-teken staat, dit %-teken verwijderen. Alle FIF's worden IF's en FDIM wordt DIM.

Toelichting:
Het programma verwerkt zowel losse gegevens, die ongesorteerd ingevoerd kunnen worden, als klasseverdelingen, die wel in volgorde dienen te worden ingevoerd.

Mediaan: Dit is de middelste waaneming uit een gesorteerde reeks.

Variatiebreedte: Dit is het verschil tussen de hoogste en de laagste waarneming.

Modus: Dit is de meest voorkomende waarneming.

Rekenkundig gemiddelde: Dit is het bekendste gemiddelde; de opgetelde waarden gedeeld door het aantal waarnemingen.

Gemiddelde afwijking: Men refereert alle waarnemingen aan het rekenkundig gemiddelde, neemt daarvan de absolute waarde, sommert deze en deelt de som door het aantal waarnemingen.

Variantie: Dit is het rekenkundig gemiddelde van de gekwadrateerde afwijkingen uit het vorige punt.

Standaard deviatie: Is de wortel uit de variantie en wordt ook wel standaard afwijking genoemd. Deze waarde wordt veel gebruikt om aan te geven in hoeverre een uitkomst van een onderzoek of enquete representatief is voor een totale groep.

Variatie koefficient: Geeft de relatieve spreiding aan, uitgedrukt in een percentage.

```
 10 REM FILENAME LOCSPR
 20 REM THEO WAAIJER
 30 DIM A(6)
 40 P.$12"**********STATISTIEK**********"'
 50 P."    LOCATIE- EN SPREIDINGS"'"         MAATSTAVEN"''
 60 P."IK BEREKEN VOOR U:"''
 70 P."   VARIATIEBREEDTE"'
 80 P."   MEDIAAN"'
 90 P."   MODUS"'
100 P."   REKENKUNDIG GEMIDDELDE"'
110 P."   GEMIDDELDE AFWIJKING"'
120 P."   VARIANTIE"'
130 P."   STANDAARD DEVIATIE"'
```

```
140 P."   VARIATIE COEFFICIENT"'
150 P.'"TIK EEN TOETS";LINK#FFE3;P.$12''
160 P."WILT U HET TOTAAL AANTAL WAAR-"'
170 P."NEMINGEN NU INVOEREN ";GOS.1000
180 B=VAL$A;FDIM%AA(B)
190 P.$12''" 1 LOSSE GEGEVENS"'
200 P." 2 KLASSENVERDELING"'
210 DO;P.'"MAAK UW KEUZE"';GOSUB 1000;D=VAL$A
220 UNTIL D>0 AND D<3;@=0
230 IF D=1;GOSUB 1200;GOTO 250
240 GOSUB 1300
249
250 REM BEPALEN VARIATIEBREEDTE
260 %A=%E-%AA(0)
270 P.$12''"DE VARIATIEBREEDTE IS..";%X=%A;GOSUBa
299
300 REM BEPALEN MEDIAAN
310 %L=(B+1)/2;L=%L
320 FIF %L=L;GOTO 340
330 %L=(%AA(L)+%AA(L+1))/2
340 P.'"DE MEDIAAN IS..........";%X=%L;GOSUBa
399
400 REM BEPALEN MODUS
410 M=0;P=0;Q=0
420 FOR N=1 TO B-1
430 FIF %AA(N)=%AA(N+1);P=P+1;GOTO 460
440 IF P>Q;Q=P;P=0;M=N;GOTO 460
450 IFP<=Q;P=0
460 NEXT N
470 P.'"DE MODUS IS............";%X=%AA(M);GOS.a
499
500 REM BEPALEN REKENKUNDIG GEMIDD.
510 %H=0
520 FOR N=1 TO B
530 %H=%H+%AA(N)
540 NEXT N
550 %H=%H/B;P.'"REKENKUNDIG GEMIDD.....";%X=%H;GOS.a
599
600 REM BEPALEN GEMIDD.AFWIJKING
610 %M=0;%N=0
620 FOR N=1 TO B;%K=0;%L=0
630 %L=ABS(%AA(N)-%H)
640 %M=%M+%L
650 %K=%L*%L
660 %N=%N+%K
670 NEXT N
680 %D=%M/B;P.'"GEMIDD.AFWIJKING ......";%X=%D;GOS.a
699
700 REM BEPALEN VARIANTIE
710 %P=%N/B
720 P.'"DE VARIANTIE IS.......";%X=%P;GOS.a
800 REM BEPALEN STANDAARD DEVIATIE
810 %S=SQR%P
820 P.'"DE STANDAARD DEVIATIE..";%X=%S;GOS.a
899
900 REM BEPALEN VARIATIE COEFFICIENT
910 %Q=(%S*100)/%H
```

```
 920 P.'"VARIATIE COEFFICIENT...";%X=%Q;GOS.a
 930 P.'''"KLAAR"'
 999 END
1000 REM INVOERCONTROL 1
1010 DO;IN.$A;M=LEN A-1;C=0
1020 FOR N=0 TO M
1030 IF A?N<48 OR A?N>57 THEN C=1
1040 NEXT N
1050 IF C=1 ;P.'"FOUT"'"NOG EENS"'
1060 UNTIL C=0;RETURN
1099
1100 REM INVOERCONTROL 2
1110 DO IN.$A;M=LEN A-1;C=0;E=0
1120 FOR N=0 TO M
1130 IF A?N<46 OR A?N>57 OR A?N=47 ;C=1
1140 IF A?N=46;E=E+1
1150 NEXT N
1160 IF C=1 OR E>1;P."FOUT"'
1170 UNTIL C=0 AND E<2
1180 RETURN
1199
1200 REM INLEZEN LOSSE GEGEVENS
1210 P.'"NU GEGEVENS INVOEREN"''
1220 FOR I=1 TO B;GOSUB 1100
1230 %AA(I)=VAL$A;NEXT I
1240 REM SORTEREN LOSSE GEGEVENS
1250 P.'"OGENBLIK,EVEN SORTEREN"'
1260 DO W=0;FOR I=1 TO B-1
1270 FIF %AA(I)>%AA(I+1);GOSUB 1500
1280 NEXT I;UNTIL W=0
1290 %E=%AA(I-1);%AA(0)=%AA(1);RETURN
1299
1300 REM INLEZEN KLASSENVERDELING
1310 F=0;G=0;H=1
1320 P.'"AANTAL KLASSEN ";GOSUB 1000;D=VAL$A
1330 FOR I=1 TO D;P.'"KLASSE "I" LOOPT"'" VAN "
1340 GOSUB 1100;%D=VAL$A;IF I=1;%AA(0)=%D
1350 P." TOT ";GOSUB 1100;%E=VAL$A
1360 P.'"AANTAL WAARN. IN DEZE KLASSE ";GOSUB 1000;F=VAL$A
1370 G=G+F;%J=((%E-%D)/2)+%D
1380 FOR J=H TO G;%AA(J)=%J;NEXT J;H=H+F;F=0
1390 NEXT I;RETURN
1399
1400aREM AFRONDEN
1410 X=%X;%Y=(100*%X)-(100*X)
1420 Y=%Y;%Y=%Y-Y;FIF %Y>.49;Y=Y+1
1430 @=0;P.X".";IF Y<10;P."0"
1440 P.Y;RETURN
1449
1500 REM WISSEL
1510 %AA(I)=%AA(I)+%AA(I+1)
1520 %AA(I+1)=%AA(I)-%AA(I+1)
1530 %AA(I)=%AA(I)-%AA(I+1)
1540 W=1
1550 RETURN
```

SOFTWARE:Speech queue Ronald Boers, Theo den Exter & Bram Poot

Met het programma Queued Speech beschik je over de mogelijkheid om op interruptbasis de SP0256 aan te sturen. Daartoe wordt een zgn. queue, een buffer, in het leven geroepen ter lengte van 256 bytes waarin de allophonen gebufferd worden alvorens ze naar de SP worden gestuurd. Terwijl deze dan aan het leuteren is, kun je verder gaan met het programma of wat je maar wilt.
Zoals je direct uit de listing kunt aflezen, worden de statements SPEAK en SAY op een gestandaardiseerde P-charme manier aangemaakt.
Het statement SPEAK wordt normalerwijze gevolgd door een adres dat het begin van de queue (256 bytes lang) vastlegt. Bij uitzondering mag in een programma (dus niet in direct mode) SPEAK zonder adres worden gebruikt in welk geval aan de DIM-pointer wordt gerefereerd. SPEAK alleen staat in feite gelijk aan DIM A(256); SPEAK A. En er mag daarna gerust verder geDIMd worden. Als er na het laatste woord niets meer gezegd hoeft te worden, kun je het queue-gebeuren (hoewel dit niet expliciet noodzakelijk is) afschakelen met SPEAK 0 (of na PROGRAM: SPEAK FALSE).
Nadat SPEAK de queue geactiveerd heeft, kan de SP0256 aan het babbelen gezet worden met het statement SAY. Achter SAY komt een willekeurig aantal argumenten die twee vormen kunnen hebben. De ene heeft de vorm van de allofoon-representatie zoals die in de bij het IC geleverde datasheet staat. Dus bv. SAY EY PA3 KK3 OR NN1 ("acorn") (zie ook Acorntjesbrood 3.1 p.36). De andere vorm bestaat uit een expressie voorafgegaan door een single quote ('). Dus bv. SAY '20 '2 '8 '58 '11. De beide vormen mogen ook gemengd worden en de spaties zijn optioneel zolang er geen misverstanden mogelijk zijn. Dus bv. SAYEY'2KK3ORNN1. In een constructie als SAY'#3 EY is de spatie nodig omdat E als hexadecimaal cijfer opgevat zou worden zonder spatie.
Als argument achter SAY kunnen als extraatje de pseudo-klanken HI, LO, NOW: en WAIT gebruikt worden. HI en LO schakelen het zevende bitje van de centronicspoort teneinde bij geschikte "spraakhardware" de intonatie van de stem te regelen. WAIT dient ervoor om in voorkomende gevallen het effect van de queue weer teniet te doen door te wachten tot de queue leeg is alvorens de interpretatie van de klanken voort te zetten. In SAY AX LL AR MM WAIT;P."!!!!!" worden de uitroeptekens pas afgedrukt als "alarm" volledig is uitgesproken.
NOW: maakt stante pede de buffer leeg, zodat hetgeen erop volgt onmiddellijk wordt uitgesproken. Bv. FOR I=0 TO 63;SAY 'I;NEXT I;SAY NOW: NN1 AW
Queued Speech is waar mogelijk behoorlijk (of liever: onbehoorlijk) gekrunchd hetgeen e.e.a. misschien ietwat onleesbaar maakt, maar waardoor het wel bijna binnen 2 geheugenpagina's past (inclusief allofoontabel).
Ten aanzien van de interrupts valt nog te vermelden dat alleen de A-poort met z'n controllijnen wordt gebruikt en wel zonder de overige instellingen van de VIA aan te tasten. B-poort, shiftregister en timers blijven dus onverkort toepasbaar. Ter demonstratie hiervan staat op het Acorntjesbroodbandje het programma KLOKJE, een real time klokje op interruptbasis (timer

2 van de VIA). Als u dit programma RUNt, verschijnt rechts boven in beeld een digitaal klokje, dat ook bij het gebruik van SPEAK en SAY doorloopt.

KLOKJE is niet helemaal netjes geprogrammeerd omdat het de oude interruptvector niet opslaat, S&S daarentegen is wat dat betreft beter verzorgd. De volgorde is dus: KLOKJE runnen, SPEAK (initieren), SAY (kletsen EN tijd aflezen).
Aangezien tijdens het SAVEn en LOADen van files alle interrupts worden tegengehouden, loopt de klok niet door en wordt het spreken onderbroken. Als de COS-operatie afgerond is, gaat het babbelen door waar het gebleven was. Een DOS-operatie echter beinvloedt geen van beiden merkbaar.

Met dit verhaal hebben we u voorlopig even genoeg verteld; het is nu de beurt aan de SP0256.

SWAP STATEMENT

Met dit statement kan men twee stukken geheugen omwisselenen. Het heeft de P-CHARME nodig voor het interpreteren van de expressies. (JSR#A4C9)
Aanroepen met: SWAP #8000,#80FF,#8100 nu wordt het bovenste-met de onderstehelft van het beeldscherm verwisseld.

```
 10 PROGRAM SWAP STAT.
 20 DIM KK(3)
 30 FORI=0TO3;KKI=-1;NEXT
 40 PRINT$21;P=A;GOSUBa
 50 PRINT$6 ;P=A;GOSUBa
 60 $T="SWAP";T=T+LEN(T)
 70 ?T=KK0/256|#80
 80 T?1=KK0%256;T?2=#80
 90 T=T+2;A=P;T!1=A
100 END
110
120a[
130:KK0JSR#A4C9;LDY@0;BEQKK3
140:KK1INC#56;BNEKK2;INC#57
150:KK2INC#52;BNEKK3;INC#53
160:KK3LDA(#52),Y
170 TAX;LDA(#56),Y;STA(#52),Y
180 TXA;STA(#56),Y;LDX@#54
190 JSR#FA0E;BNEKK1;JMP#C55B
200]
210 RETURN
```

Geschreven door:
Peter van Hengstum
Spinozahof 174
Hilversum

Software: De Intra Terrestrial Jan Biel

De Intra Terrestrial is een kleine adventure, herschreven voor de Atom vanuit VIC-20 Basic.
Om deze adventure te kunnen spelen heeft U alleen P-CHARME nodig en een 80-kolommen kaart. (Deze laatste omdat de teksten in kleine letters geprint worden.)
In deze adventure gaat U een kijkje nemen in het binnenste van onze aardbol; hier zult U ongetwijfeld het vreemde wezen I.T. tegenkomen. Luister naar wat hij U te zeggen heeft en......
Adventure ze!!

P.S. In geval van nood helpt help misschien.

Een klein voorbeeld:

```
    U bent in de BOSSEN.
    U kunt WEST gaan.
    U kunt ZUID gaan.

              ** Wat nu?Z

    U bent in de BOSSEN.
    U kunt NOORD gaan.
    U kunt WEST  gaan.

    Er is hier een WATERZAK.

              ** Wat nu?
```

FREQUENTIE

Met dit programma kunt u frequenties meten en maken.
De frequentiemeter geeft de frequentie in Hz weer, welke op de cassette poort gemeten wordt. Indien er geen signaal aanwezig is, blijft de laatst gemeten frequentie op het scherm staan. Aan de knipperende dubbelle punt kunt u zien of er gemeten wordt. U kunt de frequentiemeter b.v. gebruiken als u een bandje geleend hebt, dat niet te laden is. Meet de toon van de leader, deze moet 2400 Hz zijn. Indien nodig (en mogelijk) kunt u de snelheid van uw cassetterecorder iets bijregelen.

Met de toongenerator kunt u diverse frequenties opwekken. Indien u een frequentie opgeeft welke buiten het bereik ligt, wordt opnieuw op input gevraagd. Omdat het niet mogelijk is op precies elke freqentie te maken, wordt de dichtbijzijnste mogelijkheid berekend, en ook op het scherm afgedrukt. De opgewekte frequentie is op de cassette uitgang aanwezig, en bovendien hoorbaar in de luidspreker.

```
 10 PROGRAM FREQUENTIE-HULP
 20
 30 DIM BB(9)
 40 P.$21; GOS.p; GOS.p; P.$6
 50 P.$12"VOER IN:"'''"FREQUENTIEMETER   (F)"'
 60 P."TOONGENERATOR      (T)"'
 70 INKEY A
 80 IF A=CH"F"; G.f
 90 IF A=CH"T"; G.t
100 GOTO 70
110f@=0
120 P.$12"FREQUENTIE IN HZ"'
130 ?#E1=0
140aLINK BB0
150 X=?#81; Y=?#82
160 S=X+256*Y
170 T=6382+13*S+5*Y
180 P.$13,255000000/T"    "
190 ?#8011=?#8011:#1A
200 GOTO a
210tP.$12;INPUT"FREQUENTIE IN HZ" F
220 B=100000/F-5
230 IF ABS(F-10000/(B+5)) > (F-10000/(B+6)); B=B+1
240 ?#81=B; IF B<1 OR B>255; G.t
250 P."FREQUENTIE:"100000/(B+5)
260 P.''"DRUK SHIFT"'
270 P."NIEUWE TOON REPT"'
280 LINK BB9; G.t
290pDIM P(-1)
300[
310:BB0 LDA@0; STA#80
320     TAX; TAY
330:BB1 LDA#B002; AND@32; BNE BB1
340:BB2 LDA#B002; AND@32; BEQ BB2
350:BB3 INC#80; BEQ BB6
360:BB4 LDA#B002; AND@32; BEQ BB5
370     INX; BNE BB4
380     INY; JMP BB4
390:BB5 LDA#B002; AND@32; BNE BB3
400     INX; BNE BB5
410     INY; JMP BB5
420:BB6 STX#81; STY#82; RTS
430:BB7 LDA#80; EOR@5
440     STA#B002; STA#80
450     LDX#81
460:BB8 DEX; BNE BB8
470:BB9 LDA#B001; AND@128
480     BEQ BB7
490     LDA#B002;AND@64; BNE BB9
500     RTS
510]
520 RETURN
```

Johan Siegers

ATOM BUS DEFINITIE

De busconnector van de ATOM is nog al verwarrend, doordat er in feite twee verschillende aansluitingen worden gebruikt.
Op de ATOM-board is ook plaats gereserveerd voor twee connectoren, 1 voor aansluiting buiten de kast (PL6) en 1 voor aansluiting binnen de kast (PL7).
PL 6/7 bestaat dus uit totaal 4 rijen (tek.1) rij 1 , de A-rij van PL 6, rij 2, de B-rij van PL 6, rij 3, de B-rij van PL 7 en rij 4, de A-rij van PL 7.
We zien dus dat PL 7 het spiegelbeeld is van PL 6. Dit juist geeft de problemen over de bus.
Oorspronkelijk komt de ATOM voort uit SYSTEM, dit is een door ACORN uit gebracht EURO-RACK systeem voor laboratorium gebruik, en is bij de opkomst van de micro computer omgebouwd tot single board computer. Dit verklaart ook een andere zaak, de besturings-software van de 40*25 VDU kaart (beschreven door Aren Slootweg in AN.5 1985), want dit is de VDU kaart voor het SYSTEM.
Bij deze SYSTEM-kaarten zit de adresbus op de A-rij en is de B-rij zo goed als leeg.
Bij het ontwerpen van de ATOM-board is de 'SYSTEM' connector in de kast gezet en de aansluiting naar buiten is vanuit deze gespiegeld.
Bij het maken van de club-kaarten is van deze buiten-bus uitgegaan en is de adres-bus op de B-rij geplaatst. Dit houdt in dat deze kaarten zo in een op de buiten-bus aangesloten rack geplaatst worden, terwijl dit met de ACORN-kaarten niet mogelijk is.
Hieronder staat de buiten-connector aansluiting op een 64-polige A/B connector.

	B		A	
+5V	0	1	0	---
A15	0	2	0	---
A14	0	3	0	PB7
nWDS	0	4	0	PB6
nRDS	0	5	0	PB5
nRST	0	6	0	PB4
A8	0	7	0	PB3
A7	0	8	0	PB2
A6	0	9	0	PB1
A5	0	10	0	PB0
A4	0	11	0	CB2
A3	0	12	0	CB1
A2	0	13	0	PA7
A1	0	14	0	PA6
A0	0	15	0	PA5
D7	0	16	0	PA4
D6	0	17	0	PA3
D5	0	18	0	PA2
D4	0	19	0	PA1
D3	0	20	0	PA0
D2	0	21	0	CA2
D1	0	22	0	CA1
D0	0	23	0	---
A13	0	24	0	---
A12	0	25	0	RDY
A11	0	26	0	SO
A10	0	27	0	---
A9	0	28	0	IRQ
phi2	0	29	0	NMI
R/nW	0	30	0	SYNC
nBLK0	0	31	0	---
GND	0	32	0	GND

buiten-connector

```
-----------------------------------------------:
1 ................................ * PL6   :
2 ................................ *       :
3 ................................ : PL7   :
4 ................................ :       :
  32                             1
```

tek.1

Zelf ATOM uitbreidingskaarten ontwerpen en bouwen.

Binnen de ATOM-club zijn er verschillende mensen die een bepaald soort uitbreiding willen hebben, maar er niet aan toe komen om het te bouwen omdat er geen (bruikbare) ontwerpen gepubliceerd zijn. Of er wordt een prut-ontwerp nagebouwd wat achteraf slecht of helemaal niet blijkt te werken. Voor die mensen volgen hier enkele punten aan de hand waarvan ze zelf een ontwerp kunnen maken of een bestaand ontwerp kunnen verbeteren. Heus, het ontwerpen van een eenvoudige uitbreidingskaart (bijvoorbeeld met een PIA) is helemaal niet moeilijk. En het bouwen van een (gemodificeerd) ontwerp zonder kant en klaar print is ook best te doen.

1 HET ONTWERP

1.1 Bekijken van de mogelijkheden

Het is in eerste instantie belangrijk om na te gaan wat de uitbreidingskaart minstens moet kunnen om aan je voorwaarden te voldoen. Het verdient daarbij aanbeveling om eerst eens rond te kijken welke mogelijkheden er allemaal zijn (rondneuzen in elektronica bladen en databoeken of vragen aan iemand die er wat meer van weet). Dit kan nogal wat geld schelen, want bijvoorbeeld voor een stel I/O lijnen kan de bekende 6522 VIA gebruikt worden, terwijl een 6821 PIA van f6,50 het ook doet (wat kost die VIA ook al weer?). Mocht je de timers van de VIA nodig hebben, dan moet je die hap extra geld wel uitgeven. Zo zijn er voor andere uitbreidingen ook verschillende prijsklassen.

1.2 De ruimte

Uitgaande van een eurokaart voor een 19" rack kun je stellen dat ongeveer 4/5 van de kaart vrij te gebruiken is voor de schakeling. Dit is een zee van ruimte waar vaak meerdere uitbreidingen tegelijkertijd opgezet kunnen worden. Let wel even op het aantal lijnen dat naar buiten gevoerd moet worden: er passen bijvoorbeeld best vier PIA's op, maar wat doe je met al die I/O lijnen (96 polige connector)?. Probeer wel alles via connectoren (op de frontplaat) aan te sluiten, want anders krijg je een puinhoop. Een kaart met geheugen heeft hier natuurlijk geen last van en kan gewoon volgepropt worden.
De uitbreidingsschakeling kan ook gedeeltelijk analoog zijn, bijvoorbeeld een spraakchip met een versterker, een PIA met een stel dikke torren of een ACIA met een (telex)modem.

1.3 Aansluiting

De aansluitingen van computer-IC's zijn te vinden in databoeken en datasheets (daar is een archief voor!).
De meeste IC's hebben de volgende aansluitingen:

-datalijnen: aansluiten op de datalijnen van de ATOM.

-adreslijnen: dit zijn er meestal hoogstens vier. Ze moeten worden aangesloten op de laagste adreslijnen van de ATOM (A0-A3).

Let er bij de adres en datalijnen op dat de goede volgorde gebezigd wordt (D0 aan D0, D1 aan D1, enzovoort). Sommige fabrikanten beginnen bij 1 te tellen, in welk geval natuurlijk D1 aan D0 aangesloten moet worden, enzovoort.

-enable lijnen: hiermee kan de chip geselekteerd worden. de chip is geselekteerd als alle enable/cs lijnen logisch "1" zijn en alle enable/cs lijnen logisch "0".
De chip mag alleen geselekteerd zijn als:

*het juiste adres(bereik) geselekteerd is.

*O2 logisch "1" is.

De selektie wordt met behulp van wat logika gedaan (zie verderop).

-RD, WR of R/W lijnen: aansluiten op NRDS (RD), NWDS (WR) of R/W (R/W) van de ATOM.

-IRQ of interrupt lijnen: niet gebruiken of op de IRQ aansluiten. Alleen in heel speciale gevallen de NMI gebruiken!

Eventuele andere 'processor interface' signalen hoeven normaalgesproken niet gebruikt te worden.
Chips met gemultiplexte aansluitingen vereisen speciale truukjes waar ik nu niet verder op in wil gaan.

1.4 Selektielogica

De uitbreidingskaart moet aangestuurd kunnen worden en dus een adres(gebied) toegewezen krijgen. Let er hierbij op dat het adres nergens anders gebruikt wordt.
Je kunt uitgaan van een vast of van een instelbaar adres. In het algemeen zal een vast adres wel voldoen, ook al omdat het andere geval doorgaans resulteert in een eenmalige instelling en dus alleen maar geldverspilling is.
Na het vaststellen van het adres moet de minimaal benodigde grootte van het adresbereik bepaald worden. Bijvoorbeeld een chip met 3 adreslijnen heeft 8 adressen nodig. Voor het I/O bereik kun je ook standaard 'blokjes' van 16 adressen gebruiken, dat is wel zo overzichtelijk en er is dan ruimte zat.
Schrijf hierna de binaire waarde van het adres op een stuk papier. Bijvoorbeeld:

```
#BC4X= 1   0   1   1   1   1   0   0   0   1   0   0   X   X   X   X

      A15 A14 A13 A12 A11 A10 A9  A8  A7  A6  A5  A4  A3  A2  A1  A0
```

De X-en worden verder niet uitgedecodeerd, dus het gebruikte

geheugengebied wordt hier #BC40 t/m #BC4F.
De kunst is nu om met zo weinig mogelijk (goedkope) IC's het adres uit te decoderen. De logica kan meestal het beste zo uitgekozen worden dat er een 0 uitkomt als het adres geselekteerd is. Enkele voorbeelden staan in figuur 1. Hierna wordt O2 pas in het signaal betrokken. Het hoe en waarom wordt in 1.5 uitgelegd.

1.5 Buffering

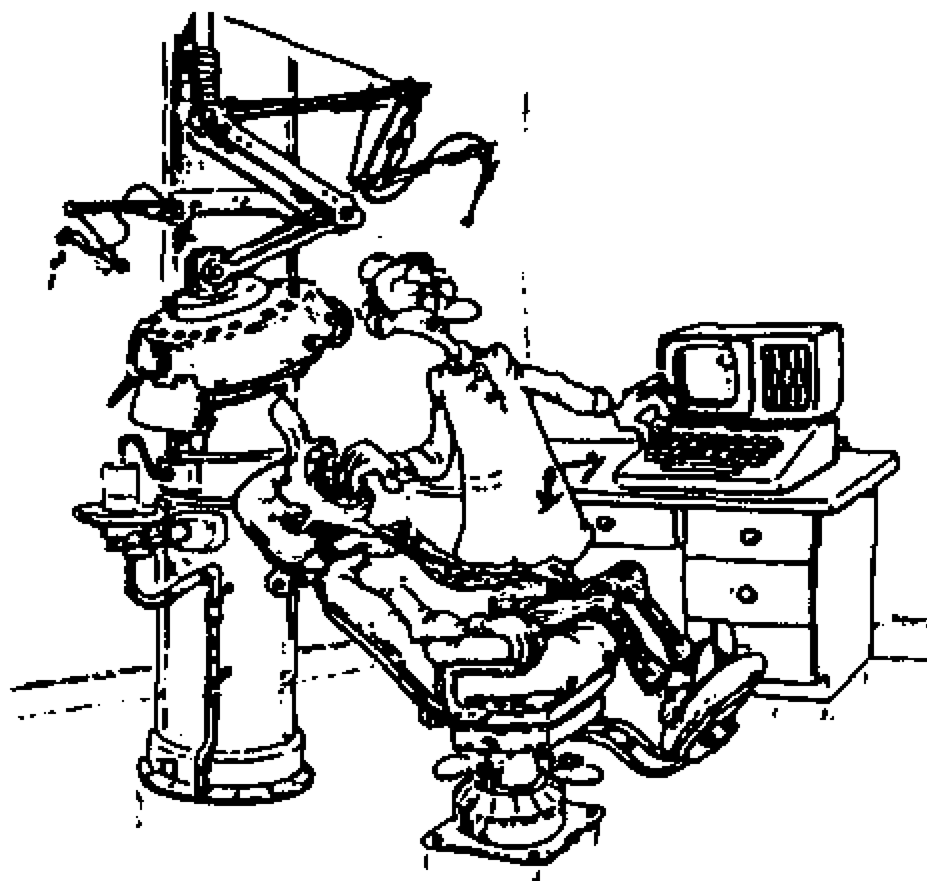
Als je (uiteindelijk) vele kaarten aan de ATOM wilt hangen dan is bufferen op elke kaart onontbeerlijk. Hiermee introduceer je wel een extra vertraging van enkele nS, maar het effect daarvan is beneden de 5 MHz of zo niet, of alleen in kritische gevallen, merkbaar.
Wat moet je bufferen?

- -Alle lijnen van de processor waar meer dan 1 a 2 ingangen aanhangen (adreslijnen, O2, R/W, enz.)
- -De datalijnen: periferie IC's zijn geen bus-drivers.

Om geen timing problemen te krijgen moeten de databusbuffers 'speciaal' worden aangesloten (figuur 2). De buffer moet ge-enabled zijn als het adresbereik geselekteerd is. Dit duurt lang genoeg om de O2 te overlappen. De IC's op de kaart zien zo altijd een geselekteerde buffer gedurende de tijd dat ze zelf geselekteerd zijn. Vandaar dat de O2 in de selektielogica apart gebruikt wordt.

wordt vervolgd

Willy Slendebroek



Extra VIA's, PIA's etc.

Naar aanleiding van enige vragen afgelopen vergadering dan hier een stukje over extra VIA's en aanverwante bouwstenen.

Standaard Atom.

In standaardconfiguratie zitten de VIA (6522) en de PIA (8255) ondergebracht in de memorymap op #B800 t/m #BBFF voor de VIA, en #B000 t/m #B3FF voor de PIA.
Dit zijn beiden 1024 bytes (1K) grote gebieden. De VIA gebruikt hiervan slechts 16 bytes (#B800-#B80F). De overige adressen zijn "spiegels" van deze 16 bytes, en zijn dus NIET vrij. We kunnen dus in principe de VIA dubbel adresseren. Het gebied #B800-#B80F is hetzelfde als resp. #B810-#B81F enz.
Ook de PIA is niet volledig uitgedekodeerd, en hiervoor geldt een soortgelijk verhaal.

Uitbreiding.

We kunnen extra VIA's enz. aansluiten op onze Atom door:

A. deze onder te brengen op vrije geheugenplaatsen, bijv. #BC00-#BC0F of #0900-#090F.

B. de bestaande VIA verder uit te dekoderen.

Op het gebied #B800-#BBFF is in principe plaats voor 64 VIA's. Standaard wordt deze ruimte volledig gevuld met slechts 1 6522. Door verdere adresdekodering kunnen we in dit gebied de extra ruimte benutten, zonder dat we de originele VIA moeten inleveren. Door uitdekoderen van bijv. 2 adreslijnen extra kunnen we hier dan 4 VIA's (3 extra) onderbrengen. Op de meest eenvoudige wijze is dit te verwezenlijken door pig-packen van een extra 74LS139. Zie voor een beschrijving hiervan Acorn Nieuws 1983 nr.4 pag.74. De 3 extra chipselectlijnen kunnen we, indien gewenst, hier dan ook vanaf halen.
Willen we de extra VIA's op een kaart in een bussysteem, dan moeten adres- en data- lijnen "buiten de bus" gebracht worden. Ofschoon dit natuurlijk zeer wel mogelijk is, ben ik toch een andere weg gegaan.

Afschaffing van de gebufferde bus.

Omdat ik in mijn Atom nogal drastisch aan het verbouwen ben geweest, kwamen er in de loop van z'n bestaan steeds ingewikkelder schakelingen in 't inwendige, die ervoor moesten zorgen dat de databusbuffer (8304, IC4) al of niet vrijgegeven werd. Eerst was dit alleen nodig voor de schakelkaart (buiten de bus brengen van Axxx, Exxx en BFFF), maar later ook Dxxx, geheugens, periferie, etc. en in de toekomst ook Cxxx en Fxxx. Om al dit plak- en bak- werk wat systematischer en mooier te organiseren, heb ik de "gebufferde bus" maar geheel afgeschaft. Hierdoor konden de extra schakelingen in de Atom eruit worden gegooid. De nadelen hiervan zijn echter:

A. De 6502 wordt iets zwaarder belast. (Geen lange flatcable op PL6/7 !)
Gelukkig is de 6502 een "taaie", en gaat dit zonder meer

goed. Ook de 65C02 had hiermee geen problemen.

B. Bij onjuiste uitbreidingen gaat de Atom helemaal plat. Vroeger kreeg je dan tenminste nog de melding "Acorn Atom".

Bij mij bleek zelfs dat de Atom in staat was op hogere kloksnelheden te draaien als voorheen (=met buffers). Als iemand weet waarom, zou ik dit graag horen.

De voordelen zijn:

A. Geen "rare" gepiggypackede extra hardware voor aansturing van het databusbuffer.

B. Eenvoudig bijbouwen van extra hardware (Schakelkaart, 32K Dramkaart etc, zijn zonder verdere interne wijzigingen te gebruiken.)

C. Rechttoe rechtaan ontwerp.

D. Vliegensvlug terug te bouwen tot standaard Atom door het verwisselen van een plugje met de originele 8304.

Om deze wijzigingen door te voeren (afschaffen van de in- en extern gescheiden bus) kunnen de adresbuffers (twee stuks 81LS95) gewoon blijven zitten. De adresbus blijft ongewijzigd, en dus ook gebufferd.

De 8304 wordt uit het voetje gehaald. In dit voetje komen twee rijen "plug". Dit plugje voert de datalijnen door naar PL6/7. Zie hiervoor het originele Atom schema. De volgende verbindingen moeten worden gelegd (IC4 voetje):

pin	naar	pin
12	-	8
13	-	7
14	-	6
15	-	5
16	-	4
19	-	1
18	-	2
17	-	3

De datalijnen zijn nu niet meer gebufferd, er bestaat dus geen binnen en buiten de bus meer!

Extra VIA ('s) op een kaart.
============================

1. Intern wordt de 6522 verder uitgedekodeerd met een '139. (A.N. '83 nr.4)
2. Of de bus wordt afgeschaft, of de nieuwe adressen worden buiten de bus gebracht. (Extra schakeling in de Atom.)
3. De extra kaart kan worden gemaakt voor PL6/7.

Op deze kaart hoeft natuurlijk niet perse een 6522 te zitten, maar 6521/6821 of andere bouwstenen kunnen ook. Meestal worden toch slechts de poorten voor eenvoudige I/O gebruikt. Ic's als de 8255 of de 6821 hebben meer "ports pro dollar", m.a.w. ze zijn veel goedkoper en eenvoudiger in het gebruik doordat ze extra's als timers en shiftregisters missen. Het verdient aanbeveling om de adresdekodering flexibel, dus programmeerbaar met draadbrugjes o.i.d., te maken. We kunnen dan later de kaart eenvoudig een andere plaats toebedelen in de memorymap. Je kunt

dan met veranderende clubstandaard compatibel blijven. Een andere suggestie is, het op een gelijke plug en pinning als PL6/7 zetten van de I/O poorten van de evt. extra VIA. Hierdoor kun je dan hardware die voor de originele 6522 gemaakt is, gebuiken op de extra VIA.

In de kast, pigpacking.

Een andere methode voor het eenvoudig bijbouwen van een extra VIA is pigypacken op de originele.
De VIA's komen dan binnen de kast, en ook binnen de bus. De busbuffers kunnen dus blijven zoals ze waren. Van de 2e VIA blijven uiteraard de poorten, chipselect, en evt. de IRQ, los. De chipselect wordt verbonden met de gepigpackede '139 (A.N. '83 nr.4).

Aldus een klein overzichtje,
Evert van Schothorst.

****

VDU-80 SOFTWARE

****

DOOR W.SALDEN

Hierbij enkele opmerkingen over de VDU-80 software:

Wanneer men op "DELETE" drukt als de cursor op het home-adres staat wordt het karakter op dit adres gewist.
Wanneer men achtereenvolgens indrukt LOCK, " cursorbesturing", COPY; springt de cursor naar een willekeurige plaats op het scherm.

De eerste fout kan voorkomen worden door het toevoegen van regel 2745.
```
2745 LDA#E0;ORA#E1;BEQ RETURN
```
De tweede fout komt door een foutieve afhandeling van de LOCK-toets.
Daarom toevoegen:
```
1695:LOCK
1696 LDA#E7;EOR@#60;STA#E7;JMP REPAET2
```

Regel 1600 #9A vervangen door "LOCK<"
Regel 1650 3e #FD vervangen door "LOCK>"
Verder heb ik de VDU-80 soft op enkele andere punten aangepast (=korter) zodat zonder de plotroutine het programma slechts #1F4 bytes lang is en de Zero Page bytes 5F en 60 niet meer gebruikt worden.

Het programma voor bovengenoemde wijzigingen staat op schijf van 2 juni "VDU-80AS".

De kortere en reeds aangepaste versie is verkrijgbaar via het bandjes archief.

VEEL SUCCES.

-0-0-0-0-0-

****
OMSCHAKELEN 80-40 KOLOMS MET ELEKTUUR VDU-KAART W.SALDEN.
****

Het is bij normaal gebruik van de Atom niet altijd nodig dat men werkt met 80 karakters op een regel. Prettiger is het om standaard 40 karakters te gebruiken.
Als men dan gaat werken met bv. de tekstverwerker kan men altijd nog omschakelen naar 80 karakters.
Dit omschakelen vereist wel een hardware aanpassing op de kaart.

BESCHRIJVING OMSCHAKELING:
==========================

Voor de omschakeing van de frequentie hebben we twee IC's nodig, een deler bv. 74LS393 en een multiplexer bv. 74LS157 men kan hiervoor een printje ontwerpen, maar stapelen op de reeds aanwezige IC's is veel eenvoudiger.
Haal pin een IC3 (74LS04) uit de socket, haal pin twee IC21 (74LS163) uit de socket.

Soldeer de pinnen zeven en veertien van het deler-IC aan de gelijkgenummerde pootjes van IC3; alle andere pinnen zover uitbuigen dat ze geen kortsluiting kunnen veroorzaken.
Soldeer de pinnen 8,15 en 16 van IC 74LS157 vast aan de dichtsbijzijnde 74LS157 op de kaart. Uiteraard weer de rest van de pinnen uitbuigen.
Leg nu de volgende verbindingen.
Pin 10 IC3 aan pin 13 IC 74LS393 en aan pin 13 IC 74LS157.
Pin 7 IC 74LS393 aan pin 12 IC 74LS393.
Pin 11 IC 74LS393 naar pin 14 IC 74LS157.
Pin 12 IC 74LS157 naar pin 1 IC3 en naar pin 2 IC21.
Nu aan pin 1 IC 74LS157 een stuk draad naar een schakelaar; "1" = hoge frequentie, "0" = lage frequentie.
Wanneer men schakelt met de VIA (6522) kan het nodig zijn de schakeldraad via een condensator met een van de voedingslijnen te verbinden.
Dit voorkomt eventuele storingen in het beeld.
De software dient nu uiteraard aangepast te worden voor het werken met 40 of 80 karakters.

PROBLEEM BIJ LAGE FREQUENTIE:
=============================

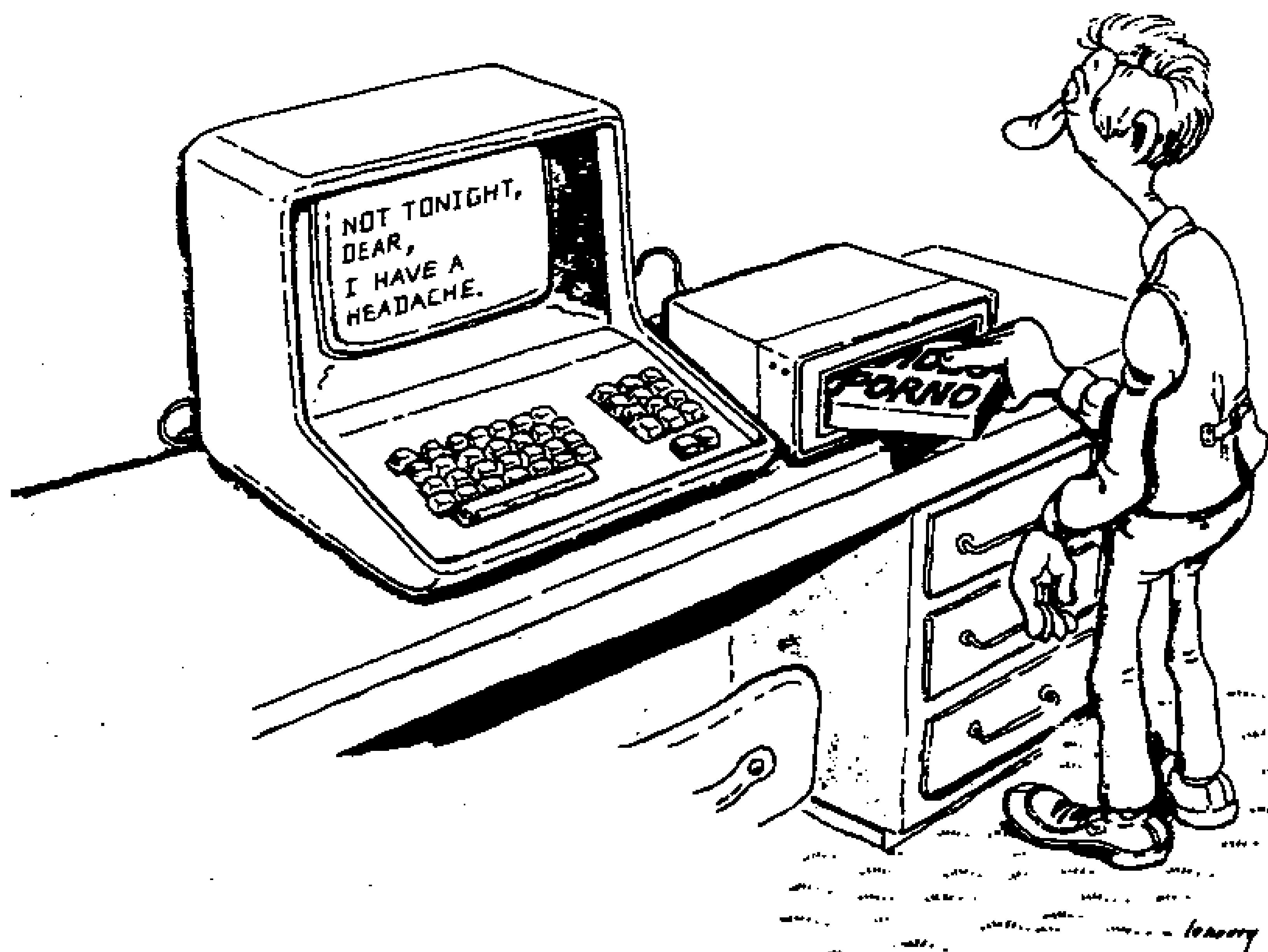
Het beeld vertoonde karakters of gedeelten hiervan die niet in het ramgeheugen aanwezig waren; wegregelen via de instelling van de monitor of de op de kaart aanwezige clockgenerator was wel mogelijk maar liet toch veel te wensen over.
De oorzaak ligt in het feit dat het clocksignaal van IC18 (74LS273) te vroeg verschijnt na het clocksignaal van de 6845; nl. tijdens het opkomen van de adreslijnen.
Door nu het clocksignaal van de 6845 te inverteren via een van de niet gebruikte poorten (N25,N15,N29,N28) is dit probleem opgelost.

OMSCHAKELEN ANTENNELIJN:

Het omschakelen tussen de videosignalen van de Atom en de VDU-kaart gebeurt het eenvoudigste met een 5 volt relais, wel een relais met een zo laag mogelijk stroomverbruik; aam het rustcontact komt het Atom-signaal, bij omschakelen wordt de VDU-kaart geselecteerd.
Wanneer men het relais nu aanstuurt met een transistor kan men ook dit omschakelen doen met een VIA 6522 of zelfs eventueel met de ongebruikte bits op de schakelkaart, hoewel dit vanwege het veelvuldig omschakelen hiervan niet aan te bevelen is.

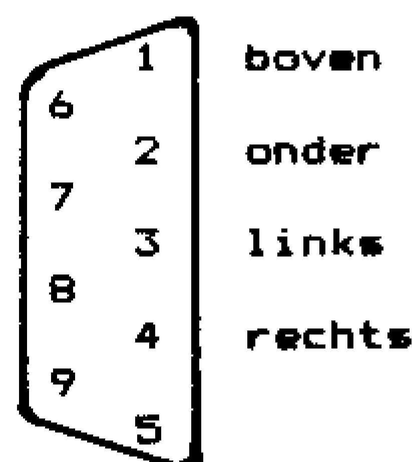
Veel succes.

-0-0-0-0-0-

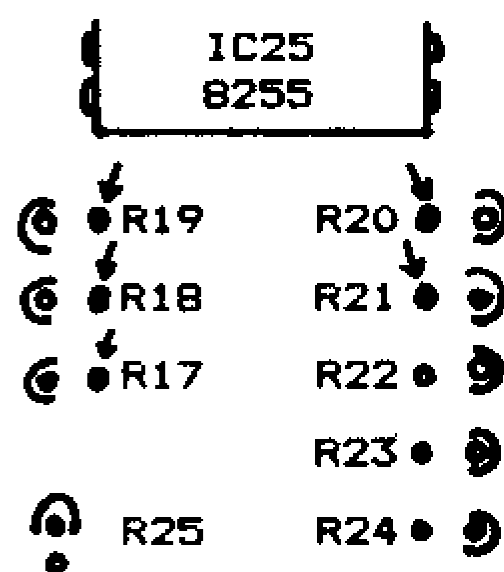


Standaardaansluiting joystick

Joysticks zijn er in twee soorten: analoge en digitale joysticks. De analoge werken met potmeters; de digitale met schakelaartjes. De clubstandaard voor het aansluiten van een joystick is alleen van toepassing op de digitale uitvoeringen; de aansluiting van analoge joysticks is niet gestandaardiseerd.
Digitale joysticks zijn allemaal van het zgn. Atari-type: vier schakelaartjes voor de vier richtingen en een vuurknop. Er is een ruime keuze in dit soort joysticks in verschillende prijsklassen: de prijzen varieren van ongeveer 20 tot 70 gulden. Alle joysticks voor Atari en Commodore computers zijn geschikt om op de Atom aan te sluiten.
Als konnektor aan de kabel van de joystick heeft Atari gekozen voor een 9-polige "vrouwelijke" D-konnektor. Hiernaast is de pinbezetting getekend (gezien vanaf de voorzijde, overeenkomstig pennummering) van de bijbehorende "mannelijke" 9-polige D-konnektor die we gebruiken om de joystick op de Atom aan te sluiten. Binnenin de joystick zijn alle 5 schakelaars aan één kant met elkaar doorverbonden. Dit gezamenlijke knooppunt vormt de retourleiding.

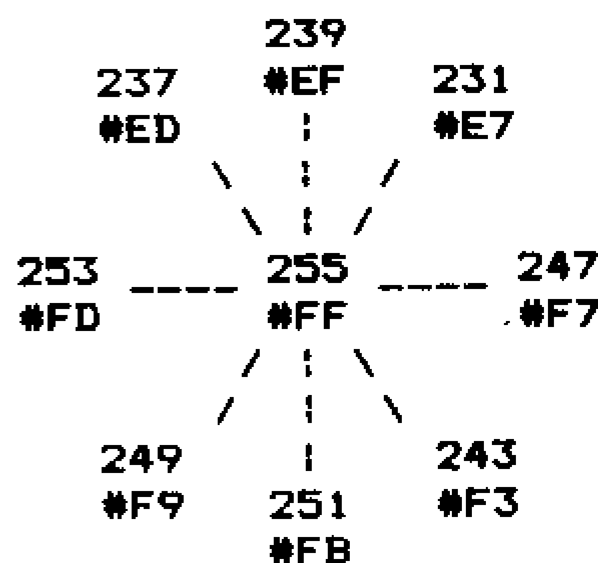


Om een joystick aan te kunnen sluiten moeten we allereerst een 9-polig mannelijk D-chassisdeel monteren op de Atom. Dit kan zowel aan de achterzijde als aan één van de zijkanten. Bij het aansluiten van de schakelaartjes in de joystick op de Atom maken we gebruik van een aantal vrije plaatsen in de toetsenbordmatrix. Het is het gemakkelijkst om aan de pullup-weerstanden van het toetsenbord te solderen. Dit gaat het beste en het netste aan de soldeerzijde van het moederbord.
In nevenstaande tekening is de situatie ter plaatse getekend. Soldeer de volgende verbindingen van de joystick-konnektor naar de weerstanden:




```
pen 1   (boven)     naar R21
pen 2   (onder)     naar R19
pen 3   (links)     naar R18
pen 4   (rechts)    naar R20
pen 6   (drukknop)  naar R17
pen 8   (retour)    naar IC26 pen 1
pen 5,7 en 9 zijn niet aangesloten.
```

N.B. Denk eraan dat je aan de goede kant van de weerstanden soldeert, anders krijg je kortsluiting!
Nu kunnen we op geheugenlocatie #B001 (B-poort van de 8255) de stand van de joystick uitlezen. In de tekening hiernaast is aangegeven welke waarden we krijgen voor de 8 mogelijke richtingen, zonder ingedrukte drukknop. Met ingedrukte knop worden alle getallen met 1 verminderd.



Als we twee joysticks willen aansluiten gaan we in principe net zo te werk; alleen moet de retourleiding van de tweede joystick nu niet naar IC26 pen 1, maar naar pen 2 van IC26. De overige aansluitingen van de tweede joystick zijn precies gelijk aan die van de eerste.

Ronald Boers

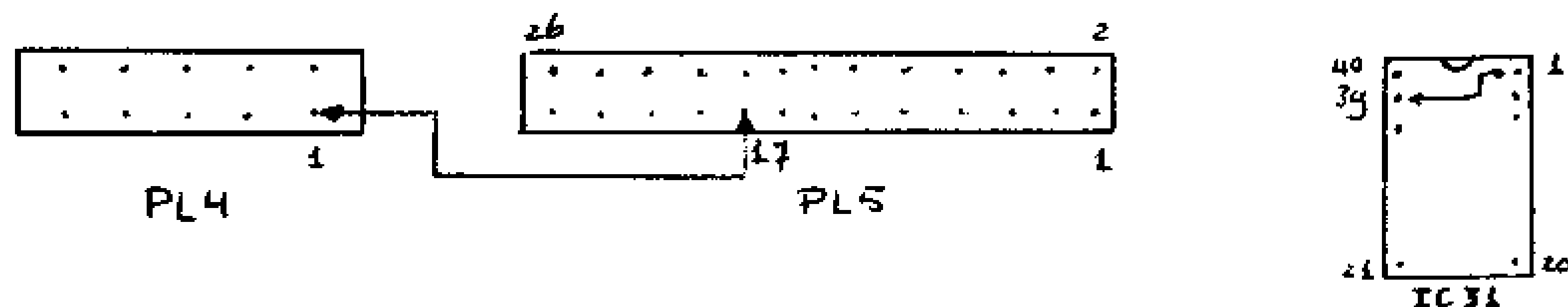
STANDAARDOMBOUW 8E-PRINTERBIT

In het kader van de beschrijving van een standaard ATOM volgt hier de beschrijving van de ombouw van de normale printerinterface naar een 8 bits printerinterface. De normale printerinterface wordt gevormd door de IC's 1 en 50 op de moederprint. Van IC1 (6522: VIA) wordt de lijn PA7 van de printerpoort gebruikt om de BUSY lijn van de printer uit te lezen. Hierdoor blijven van poortA van de VIA 7 bits over voor dataoverdracht naar de printer. Voor de normale ASCII karakters is dat voldoende, maar wanneer we (met een geschikte printer) grafische plaatjes willen printen in de bit-image mode van de printer komen we 1 bit (het achtste) tekort.

Deze kan er echter eenvoudig aan toegevoegd worden en wel als volgt:

- een verbinding leggen tussen pen1 van PL4 (video conector) en pen17 van PL5 (printerconnector)
- om ervoor te zorgen dat dit bitje nu ook naar de printer gaat moet link LK5 op de ATOM print omgelegd worden. Dat betekent dat pen39 van IC31 (6847: CRT-controller) aan 0 Volt gelegd moet worden (bv. aan pen1 van IC31) en dus de bestaande verbinding tussen pen39 van IC31 en pen1 van PL4 doorgekrast moet worden.

In onderstaande figuur is een en ander nog eens aangegeven. (gezien vanaf de soldeerzijde van de ATOM print)



Het resultaat van deze ombouw is een 8-bits printerinterface waar echter wel extra software voor nodig is.

Opmerking:

Binnen de club zijn diverse dump-programma's aanwezig voor allerlei verschillende printers (hoofdzakelijk voor het dumpen van een mode4 plaatje op de printer). Over het algemeen werken die programma's alleen goed in combinatie met die printer waarvoor ze oorspronkelijk geschreven zijn.

Jan Biel

Je kent ze wel: die kastjes van een paar honderd gulden, die je tussen je computer en je printer kunt hangen. De computer schrijft erin en de printer leest er in zijn eigen (trage) tempo weer uit. Printerbuffers, met als voordeel dat computer (& eigenaar) tijdens printroutines niet meer 95% van de tijd moeten wachten. Nadeel is natuurlijk wel de prijs en als je dan nog bedenkt dat zo'n buffer niet veel meer is dan een geheugen, dan ligt het
idee om de computer zelf de bufferfunktie te laten verrichten wel voor de hand.

Het programma PBUFFER gebruikt een op te geven geheugen gebied als bufferruimte. De computer begint hierin te schrijven en wanneer de "bovenkant" van de bufferruimte bereikt is gaat hij weer bij de "onderkant" verder. De bufferruimte is als het ware een cirkel geworden. De printer leest uit deze cirkel en probeert de computer in te halen. Zo rennen printer en computer in een cirkel achter elkaar aan. Zie je 't voor je? Wanneer de computer zo snel is dat hij een volle ronde voorloopt op de printer en de buffer dus vol is, dan wordt dit aangegeven door een knipperende linker bovenhoek op het scherm.

De printer wekt hier met zijn ACK-lijn zelf een interrupt signaal op zodra hij een volgend karakter wil lezen. Daardoor kan de computer wanneer hij zijn te printen karakters in de buffer heeft gezet gewoon verder gaan met het volgende karwei. De printer zoekt het verder "zelf" wel uit.

PBUFFER kan niet alleen voor programma listings worden gebruikt, maar ook voor uitvoer van rekenresultaten, assembler uitdraaien enz. enz. Nadat PBUFFER geinitialiseerd is kan, wanneer er iets geprint moet worden, de buffer geopend worden m.b.v. PRINT$2 (of CTRL B), sluiten gaat met CTRL C (of PRINT$3). Wanneer wij de buffer sluiten, maar de buffer nog niet leeg is gaat de printer gewoon door met uitlezen. De computer zal nu niet in de buffer schrijven, tot dat deze weer geopend wordt met PRINT$2.

Wanneer het PBUFFER programma geassembleerd is naar bijv. #3EC0 en we als bufferruimte het gebied #9800 tot #9E00 willen gebruiken dan initialiseren we met

```
X=#98;Y=#9E;LINK#3EC0
```

Het is natuurlijk ook mogelijk om van PBUFFER een P-CHARME statement te maken. In dat geval moeten regels 10 t/m 60 vervangen worden door:

```
10 PROGRAM PBUFFER.STAT
20
30 DIM LL14;F.N=0TO14;LLN=-1;N.
40 P.$21;P=A;GOS.a;P.$6;P=A;GOS.b
50 $T="PBUFFER";T=T+LENT;?T=A/256|#80;T?1=A%256
60 T?2=#80;T=T+2;A=P
```

Regel 190 moet worden vervangen door

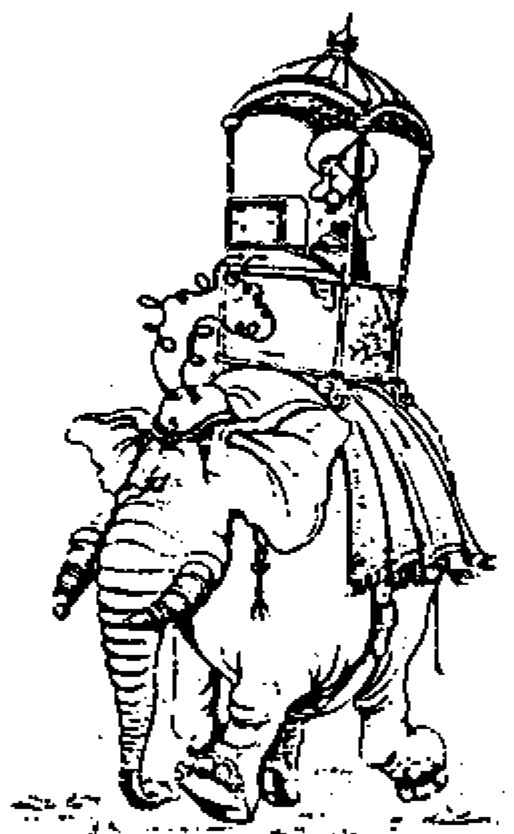
```
180 JSR#C78B;JSR#C231;JSR#C4E1
190 LDA#16;STA S;LDA#17;STA B;LDA@0;STA#4
```

Verder moet de RTS in regel 240 vervangen worden door een JMP#C55B.
Initialiseren gebeurt dan bijv. met het commando PBUFFER #98,#9E.

hans hegt

****
** TIP **
****

oor de moeilijk verkrijgbare IC 81LS95 is een vervanger nl. de 4LS465. (Is tevens goedkoper).



* * * r t t y - i n f o * * *

Informatie voor gebruik van het rtty-programma.

((made by PE1HMQ 1983 v1))

Hardware

Voor gebruik van het programma moet de standaard Acorn atom uitgerust zijn met het volgende geheugen:
#2800 - #3BFF (=ic10-ic19)
#8400 - #97FF (=ic32-ic41)
Zie hiervoor: Atom technical manual.
Met het programma kan men RTTY (=telex) ontvangen en uitzenden. Wordt het programma alleen voor ontvangen gebruikt dan hoeft men alleen een telex-converter aan te sluiten. De converter moet een uitgangssignaal geven op TTL-niveau. Voeding van de converter kan via PL6 uit de computer. Schema's van telex-converters kan men vinden in de diverse elektronica hobby bladen, bijv.:elektuur 1982 8-14, elektuur 1983 6-58.

Voor het uitzenden van RTTY zijn de volgende verbindingen van belang:
1. SQUELCH (ingang)
2. PTT (uitgang)
3. AFSK (uitgang)
4. Call-gever (uitgang)

Squelch: Deze ttl-ingang wordt verbonden met het squelch-signaal van de ontvanger. Met een comparator is dit signaal eenvoudig op TTL-niveau te brengen. Doel van deze verbinding is het voorkomen van per ongeluk zenden en het mogelijk maken van automatisch CQ geven. Is het niet mogelijk de squelch aan te sluiten dan moet deze ingang met 0v verbonden worden en vervalt de CQ-mogelijkheid.

PTT: Deze uitgang dient voor het omschakelen van zenden en ontvangen van de aangesloten zend-ontvanger. Ter voorkoming van zenden bij uitgeschakelde computer voeding worden twee transistors tussen geschakeld (zie schema)

AFSK: Deze uitgang geeft direct de voor telex benodigde tonen. Alleen een verzwakker en filtering is nodig.

Call-gever: Ter identificatie van het amateurstation is het verplicht regelmatig tijdens de uitzending de roepnaam uit te zenden. Via de callgever uitgang is het mogelijk de roepnaam in morse uit te zenden. Het morse signaal wordt na verzwakking en filtering aangeboden op de modulatie ingang van de zender.

Aansluiting van de I/O-poorten:

PB0 - RX-converter ingang (mark='1', space='0')
PB1 - Squelch ingang ('1' bij signaal)
PB5 - Call-gever uitgang

PB6 - Ptt uitgang ('0'=tx,'1'=rx)
PB7 - Afsk uitgang

Overige pennen van poort B worden niet gebruikt.
Poort A wordt voor de printer gebruikt.

Onderdelenlijst

T1 = BB547 (of andere NPN tor)
T2 = T1
R1 = 10k
R2 = 6k8
R3 = 22k instelpotmeter
R4 = R3
R5 = 10k
R6 = 10k
R7 = 15k
C1 = 15n
C2 = 120n
c3 = 15n

Software

Algemeen:

* Indicatie van de mode in de rechterbovenhoek van het scherm,
  r = ontvangst (rx)
  t = zenden (tx).

* Indikatie van aantal ontvangen of verzonden karakters.

* Auto-screen-edit zorgt voor een goed leesbare tekst (voorkomt halve woorden aan het einde van de regel).

* Stabiele computer-made mark- en space frequenties. Deze frequenties kunnen eenvoudig aangepast worden door in de eerste regel van het basic deel andere frequenties in te vullen. Zorg er wel voor dat deze regel net zo lang blijft.

* De callgever kan voordat met 'run' begonnen wordt eenvoudig met de volgende instructie geprogrammeerd worden:
  $#34F2="uw roepnaam" Maximaal 12 karakters !!!

Het 'runnen':
Na het laden van het programma wordt gestart met de instructie 'run'. Hierna vraagt het programma om de te gebruiken baudrate. Voer dit altijd als een integer in, b.v. 45.45 baud wordt ingevoerd als 45 of 46.
Gebruikelijke snelheden zijn: 45.45
50
75
110 baud.
Na het geven van 'return' verschijnt het selectie menu. het selectie menu omvat:
R = RX
T = TX
Q = CQ

B = Baudrate
M = Memory
D = Display memory
E = Enter text
L = Load memory
S = Save memory
P = Printer
I = Test input

Na het kiezen van een onderdeel wordt teruggekeerd naar het menu door het indrukken van de 'esc'-toets (m.u.v. b, m, l en s).
RX:
Het ontvangen signaal wordt omgezet en weergegeven. Is er een geheugen gekozen, dan wordt de tekst ook hierin opgeslagen. Speciale symbolen zijn:
 b = Bell
 f = Fout (karakter bestaat niet).

TX: Uitzenden van tekst uit geheugen of rechtstreeks van af toetsenbord. Is er een geheugen gekozen dan wordt direkt de hierin aanwezige tekst uitgezonden. Bij het einde van de tekst (of na 'esc') wacht het programma op tekst vanaf het toetsenbord. Is geen geheugen gekozen dan kan dit alsnog gedaan worden door intoetsen van m (shift-M). De call-gever wordt aktief na intoetsen van c (shift-C).

CQ: de tekst in een van de geheugens wordt uitgezonden waarna overgeschakeld wordt op ontvangst. Wordt gedurende enige seconden geen signaal ontvangen (squelch ='Ø'), dan wordt opnieuw de tekst uitgezonden. Wordt er wel signaal ontvangen dan wordt de cq-procedure gestopt en blijft het programma ontvangen.

Baudrate: Wijzigen van de baudrate. Eventueel kan met de 'esc'-toets het programma gestopt worden om de mark- en space frequenties te wijzigen. Dit is ook tijdens het zenden mogelijk (!).

Memory: Het programma geeft de mogelijkheid om tekst op te slaan in 3 geheugens. de grootte van de geheugens is:
 1 - 256 karakters
 2 - 1792 karakters
 3 - 5632 karakters.
Wordt Ø gekozen dan zijn de geheugens buiten gebruik. Automatische uitschakeling na zenden of ontvangen voorkomt het overschrijven of per ongeluk uitzenden van tekst in een geheugen. Mensen met een 16 K kaart kunnen de grootte van geheugen 2 vergroten door voorafgaand aan het runnen op adres #2F33 het einde van het geheugen aan te geven (Alleen hoogste byte). Dus moet geheugen 2 lopen tot #6000, voer dan in: ?#2F33=#60.

Display memory: De inhoud van het gekozen geheugen wordt weergegeven. Tijdelijk onderbreken is mogelijk door indrukken van de shift-toets.

Enter text: Opslag van in te toetsen tekst in het gekozen geheugen. Dit deel van het programma is geen uitgebreide tekstverwerker. Om toch fouten in eerder ingetoetste regels te

kunnen herstellen wordt na 'return' op de zelfde regel verder gegaan. In de tekst opnemen van c (=shift C) zorgt ervoor dat bij uitzenden van het geheugen de callgever geaktiveerd wordt. Koppelen van geheugens is mogelijk door intoetsen van m (=shift M) gevolgd door het er na uit te zenden geheugen. Ook kan het nummer van het geheugen zelf worden ingevuld. Tijdens uitzenden wordt dan steeds dezelfde tekst herhaald. m is a.h.w. een goto instruktie.

Load memory: Laden van geheugen vanaf tape. het laden en saven van geheugen gebeurt met de zelfde baudrate als welke aanwezig is voordat het programma gestart is (300 of1200 baud). De tekst wordt in het oorspronkelijke geheugen teruggeladen. Voor een overzicht van de tekstfiles (=catalog) een niet bestaande (of geen) naam invoeren. Behalve de naam wordt ook het geheugennummer weergegeven. Het laden kunt u onderbreken met de 'ctrl'-toets.

Save memory: Wegschrijven van de tekst in het gekozen geheugen naar tape.

Printer: in- of uitschakelen van de printer.

Test input: Snelle kontrole van de ingangen. Het 'hoog' of 'laag' zijn van de ingangen wordt symbolisch weergegeven.

Het programma moet worden geladen op adres #2900. Het eindadres wordt gevonden met de instructie: A=#34F2; PRINT&(A+LEN A+2). Het programma kan nu gesaved worden. De eventueel ingevulde roepnaam voor de callgever wordt gelijk mee gesaved. Gebruik: *SAVE"RTTY"2900 (eindadres) C2B2. Voor (eindadres) wordt het eerder gevonden eindadres in hex-vorm ingevuld. Het programma kan daarna met 'LOAD' weer geladen worden. Om fouten te voorkomen verdient het aanbeveling om eerst een tijdje 'droog te oefenen' alvorens men echt gaat uitzenden.

Veel succes met RTTY, good dx and 73 from PE1HMQ

hoezo, onleest

```
10 REM hoezo, onleesbaar?
20 Q=#2800; P=Q; X=#80
30[LDY@X; STY X+1
40 LDX@0; STX X
50 STA (X,X)
60 INC X \ niet INX!
70 BNE P-4
80 RTS
90]INPUT"een getal graag"A
100 LINK Q
110 END
```

Ombouw programmer.

Omdat ik laatst een stel 2732's tegen een zeer aantrekkelijke prijs over kon nemen, besloot ik m'n programmer hier ook maar geschikt voor te maken ...
Bestudering van Acorntjes Brood 2.4 leerde dat het niet moeilijk kon zijn. Alleen die oplossing met al die losse schakelaars (wie heeft er nu een drie-polig omschakelaar van redelijk formaat in de kast liggen?), zag ik niet zo zitten.

Dan zelf het probleem nog maar eens bestuderen. De 2732 en de 2532 verschillen kwa aansluitingen op drie punten. Het betreft de aansluitpennen 18, 20 en 21. 2532 plaatst hier achtereenvolgens A11, CE en Vpp, terwijl 2732 het met CE, OE/Vpp en A11 doet.
Dit naast het feit dat er in principe 6 (zes) mogelijke situaties ontstaan, te weten:

- 2532 lezen (25R)
- 2532 programmeren 21V (25P21)
- 2532 programmeren 25V (25P25)
- 2732 lezen (27R)
- 2732 programmeren 21V (27P21)
- 2732 programmeren 25V (27P25)

Uit deze gegevens laat zich eenvoudig een aansluittabel voor beide types EPROM en alle situaties distilleren. Deze kan er dan als volgt uit zien:

pen \	25R	25P21	25P25	27R	27P21	27P25
18 \	A11	A11	A11	CE	CE	CE
20 \	CE	CE	CE	low	21V	25V
21 \	5V	21V	25V	A11	A11	A11
stand	1	2	3	4	5	6

En wat is er nu makkelijker om het geheel omschakelbaar te maken met een enkele schakelaar (alles hardware matig zo ombouwen, dat het software matig te schakelen is, hoor ik iemand zeggen ...). De keus valt hiermee op een draaischakelaar. En wel een met minimaal zes standen en minimaal 3 p-kontakten.

Nu weten we dus exact wat we op de pennen 18, 20 en 21 aan moeten bieden, om het beoogde te bereiken. Voor de ombouw krassen we de koperbanen op de programmer print vlak bij het EPROM voetje door bij de genoemde pennen. Aan de genoemde pennen komen nu de moeder (p) kontakten van de draaischakelaar. De schakelkontakten worden dan volgens de tabel aangesloten.

Tip: koop bij de draaischakelaar een knop met een standenaanduiding, zodat steeds duidelijk blijft in welke stand de programmer staat.

Wibo Visser.

SOFTWARE:Epromprogrammeerprogramma Bram Poot

Voorgeschiedenis.

Toen ik destijds de epromprogrammeerdienst van de regio vervulde, was de Josbox zo goed als het enige "programma" dat ik voor onze leden heb mogen programmeren in een eprom. Aangezien dit een compleet pakket is ter lengte van exact 4k, was het basic-programmeerprogramma dat in het engelse computerblad Practical Computing bij het schema van de programmer stond, goed bruikbaar. Het was alleen een beetje traag: verifieren en copieren duurde maar liefst ongeveer 3 minuten.
Ik was dan ook bijzonder gecharmeerd van het programma PROM 2.0 van Maarten van Alphen dat e.e.a. in machinetaal uitvoert. Bovendien zitten daar nog faciliteiten als een blank check en een sum check in. Ikzelf heb daar nog een "program check" bijgemaakt die controleert of een programma "over de eprominhoud heen" gezet kan worden. Een eprom hoeft nl. niet per se leeg te zijn, getuige het volgende plaatje:

te programmeren bit	eprominhoud	kan geprogrammeerd worden?
0	0	ja
0	1	ja
1	0	neen
1	1	ja

Als situatie 3 zich niet voordoet, kan de eprom ondanks dat-ie niet leeg is, toch geprogrammeerd worden. Het met deze faciliteit uitgeruste programma PROM 2.0X wordt nu intensief gebruikt door de huidige epromprogrammeerdienst die in zijn nog korte bestaan er reeds een stuk of 150 eproms mee heeft behandeld (en niet alleen Josbox!).
Na het verschijnen van het duitse computerblad C't (nr.2 1985) waarin een zeer uitgebreid artikel over eproms en het programmeren ervan staat, vond ik dat ik maar eens een geheel nieuw epromprogrammeerprogramma moest schrijven, waarin op eenvoudige wijze een aantal interessante faciliteiten in te bouwen zijn.
Waarvan akte.

Overzicht.

Het programma biedt het volgende:

- alle klassieke faciliteiten, zoals
  - blank check
  - read eprom
  - write eprom
  - sum check
  - verify check
- overwrite check (de hierboven beschreven faciliteit)
- definieren van het werkgeheugen
- definieren van een deelgebied van het werkgeheugen
- programmeren op interactieve wijze (byte voor byte)
- bepalen van de CRC check

Opstarten.

Als u het programma aanlinkt, wordt het scherm schoongemaakt en verschijnt bovenaan een dubbele statusregel. Een aantal standaardwaarden wordt daar ingevuld. Dit zijn:
- ondergrens van het deelgebied ("LOW") = #000
- bovengrens van het deelgebied ("HIGH")= #FFF
- startadres van het werkgebied ("MEM") = #7000

Het gebied tussen LOW en HIGH (inclusief deze grenzen) wordt voor de meeste commando's als referentie gebruikt. In b.v. de SCREENROM is het gebied #AD00 - #AFCF leeg. Als ik dus LOW op #D00 en HIGH op #FCF instel (zie verderop) dan krijg ik bij het commando B de melding "blank check ended" zonder dat er "fout"-meldingen zijn opgetreden. De totale ROM is niet leeg, het als voorbeeld gestelde gebied wel, vandaar.
ADDRESS en ?ADDRESS geven resp. het adres en de inhoud van dat adres aan van het laatst gelezen of geschreven byte van de eprom. I.h.a. is dit adres gelijk aan HIGH, maar als u b.v. escape heeft ingedrukt kan dit ook een ander adres zijn.

De commando's.

Als u een overzicht van alle commando's wilt hebben, tikt u "?" in. Nadat u kennis heeft genomen van de aanwezige mogelijkheden, drukt u op return waarna u een commando kunt invoeren.
De waarden in de tweede statusregel kvnt u op elk moment wijzigen met L, H en M voor resp. LOW, HIGH en MEMORY. Na wijziging van L en H wordt gekeken of HIGH kleiner is dan LOW. In dat geval moet u een nieuwe waarde invoeren.
De volgende commando's werken alleen op het deelgebied.

B - blank check, alle bytes moeten #FF zijn. De bytes die hier niet aan voldoen worden gelist.
R - read eprom, leest de bytes in in het overeenkomstige geheugengebied.
O - overwrite check, bekijkt of het geheugen over de eprominhoud geprogrammeerd kan worden. De bytes waarvoor dit niet kan, worden gelist.
V - verify check, vergelijkt de bytes met het overeenkomstige geheugen. De "verkeerde" bytes worden gelist.
P - program eprom, programmeert de eprom op klassieke wijze. Dit duurt ongeveer 3 1/2 minuut voor 4k.
S - sum check, berekent de som van de bytes en drukt het resultaat in 2 bytes af.
C - crc check, berekent de crc van de bytes en drukt het resultaat af (crc volgens signature programma in handboek p.93).

Dan is er nog de mogelijkheid om afzonderlijke bytes te programmeren via het I-commando. U wordt gevraagd een adres te specificeren (default: #000), waarna u de inhoud ervan vermeld krijgt. Vervolgens mag u een nieuwe waarde opgeven die dan in de eprom wordt geprogrammeerd. In de eerste statusregel wordt altijd de werkelijke eprominhoud afgedrukt, zodat u daar kunt controleren of het gelukt is.
U kunt het I-commando verlaten met escape.

Als u klaar bent met programmeren kunt u het programma verlaten met (uiteraard) een keiharde BREAK, maar netter is via Q (QUIT) Na de vraag "are you sure?" ("bent u er volkomen zeker van dat u het programma wilt verlaten?") tikt u dan de letters "Y", "E" en "S" in.
U kunt overigens elk commando afbreken met een escape. Sommige routines gaan echter zo bloedje snel dat u daar nauwelijks tijd voor heeft. Tijdens het programmeren zou het weleens zinvol kunnen zijn. Denk dan wel aan de READ/PROGRAM-schakelaar!!!

Opmerkingen bij het programma.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Alleen in de regels die gemerkt zijn met het label p wordt gebruik gemaakt van P-charme op #1XXX. Heeft u dit niet dan dient u deze regels aan te passen en aan het begin (b.v. na .list) .option :0100 0000 toe te voegen, of, beter nog, BRANQUART-SOS te implementeren.

Het programma maakt voor de timing van de programmeerpulsen gebruik van timer 2 van de VIA. Dit lijkt me geen probleem op te leveren, aangezien u de VIA toch al nodig had vanwege haar I/O-poorten. Het programma mag dus nog steeds zonder meer geinterrumpeerd worden en draait ook goed bij degenen met een 1/2 MHz fly back puls.

De vertaalde code is relatief nogal lang (8 geheugenpagina's). Hier is slechts 1 reden voor en dat is het gebruik van nogal wat tekst, hetgeen echter de handelingen naar mijn mening erg duidelijk maakt.

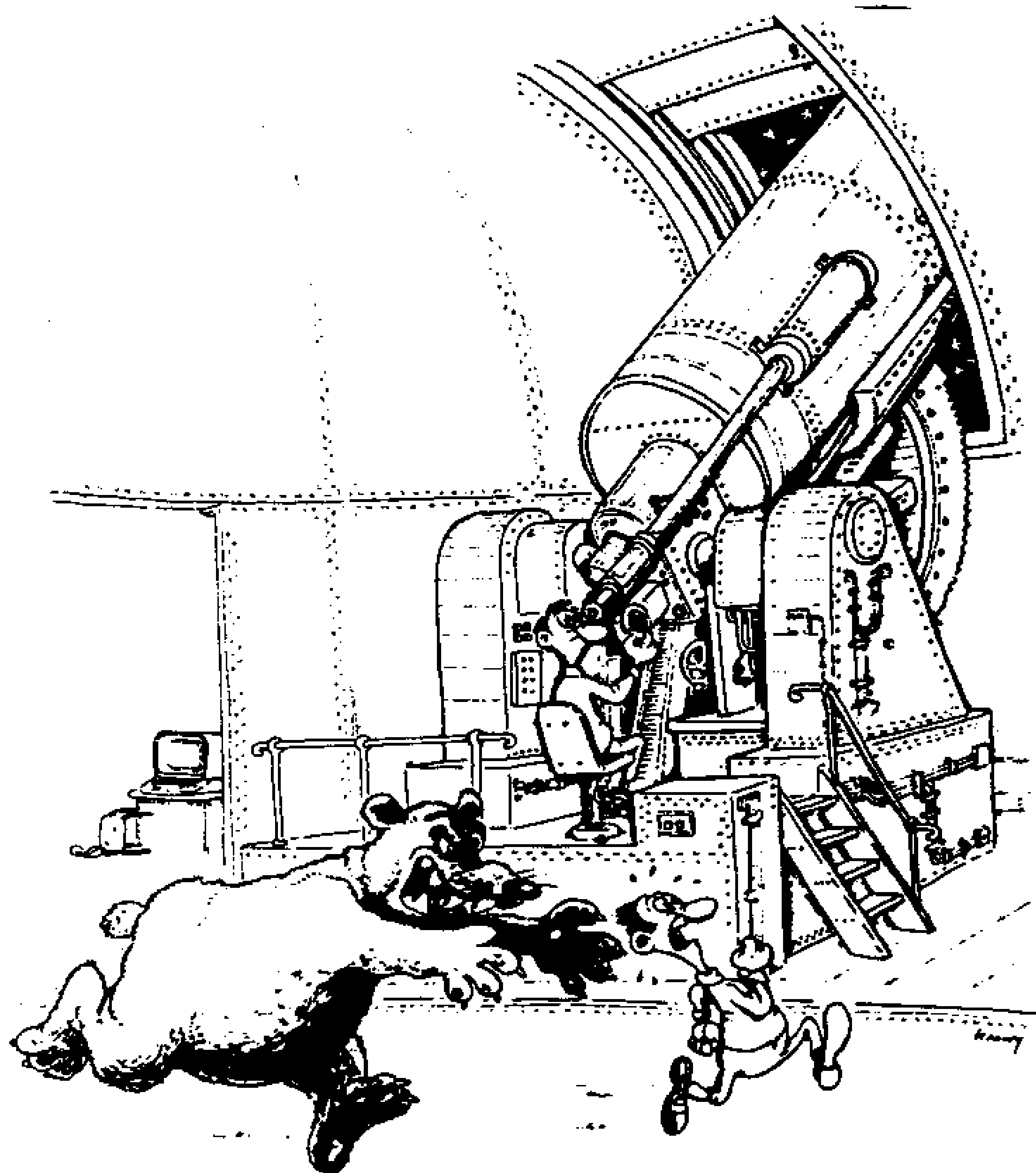
Vaak wordt de commandouitvoer gestopt om u de gelegenheid te geven deze te lezen. U wordt in zo'n geval geacht op de returntoets te drukken om het commando te continueren.

Ikzelf bezit naast het normale Atom-beeldscherm een 80-koloms-VDU. Het programma moest op beide zonder meer kunnen draaien. Het probleem daarbij is dat vanwege de programmerhardware de printer driver verwijderd moet worden, en dat is niet universeel mogelijk. De meest universele methode die ik kon bedenken staat beschreven in het handboek op pag.170. Deze methode heeft tot gevolg dat dit programma ALLEEN draait met beeldschermroutines waarvan de write character routine begint met JSR #FEFB en gedurende de sessie niet wordt veranderd. De combinatie met SCREENROM is dus b.v. niet mogelijk. Bij twijfel zult u het originele scherm moeten gebruiken.

De programmer heeft een adresbus met een wrap around ter lengte van 12 adreslijnen. Dit wil zeggen dat bv. adres #1234 hetzelfde is als #0234. Als u de bovengrens ("HIGH") dus op #1000 instelt, wordt adres #000 bv. bij "PROGRAM" tweemaal geprogrammeerd met i.h.a. verschillende waarden. De geheugenpointer is nl. wel degelijk 16 bits breed.

Het programma gebruikt de zero page bytes #80 t.e.m. #8D.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Het programma voorzag oorspronkelijk ook in de mogelijkheid van het programmeren van een eprom op geavanceerde wijze. Deze methode, een algoritme dat INTEL heeft ontwikkeld, zou een eprom van 4k in ongeveer 1.5 tot 2 minuten kunnen programmeren i.p.v. de klassieke 3.5 minuut.
Edoch, helaas, het algoritme is ontwikkeld voor eproms van 8k, 16k etc. en het was een ietwat simpele veronderstelling te denken dat het ook zou werken voor "verouderde" types. Het blijkt nl. zo te zijn dat de eprom tijdens de programmeerperiode, dus met de schakelaar op "PROGRAM", gelezen moet kunnen worden en dat kan alleen met de grotere eproms.



*I forgot to type "Find the **constellation** big bear" . . .*

De volgende twee programma's zijn bedoeld om het bekende 'bingo-molentje' te vervangen. Na elke druk op een willekeurige toets zal de Atom een getal 'bedenken', ook wordt er bijgehouden welke getallen al geweest zijn. Het eerste programma genereert getallen tot en met 75, het tweede programma geeft getallen tot en met 90, dit omdat het spel op twee manieren schijnt te worden gespeeld.

Na het 'runnen' worden de getallen 1 t/m 75 (of 1t/m 90) random weggezet in het geheugen waardoor we ze later allemaal even snel op kunnen halen, anders zouden we mogelijk op 't laatst wat al te lang moeten wachten alvorens er een nieuw getal was gegenereerd.

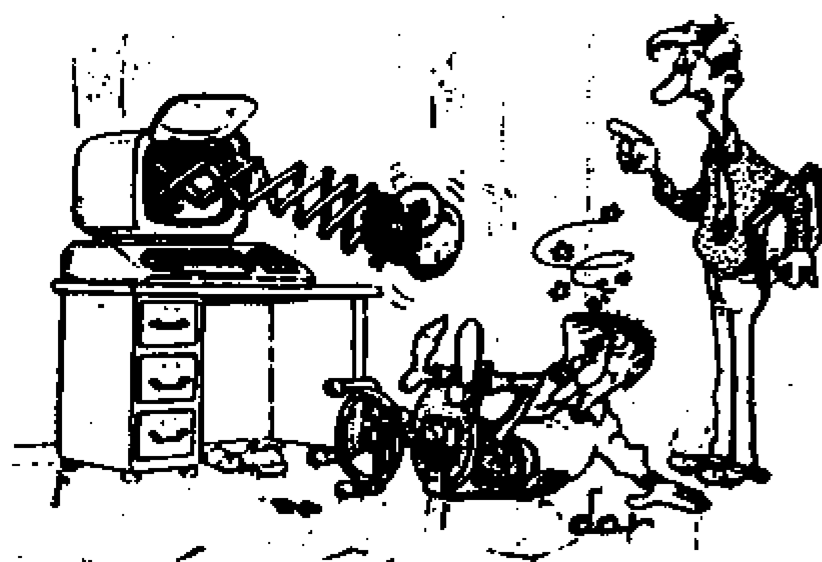
Misschien dat de bezitter van een 'speech synthesizer IC' de Atom zelf de gekozen getallen op kan laten lezen.....

J.P. Tap.

```
 10 PROGRAM BINGO 75
 20 REM J.P.TAP. UITHUIZEN
 30 REM TEL.NR. 05953-2177
 40 P.$12;?#E1=0;S=#8001;W=#FE94;@=2
 50 DIM NN(75)
 60 P."  b  i  n  g  o  "
 70 F.G=0 TO #1F0 S.#20
 80  F.X=#8022+G TO #802F+G S. 3
 90  ?X=#2E
100  N.X
110 N.G
120 F.G=1 TO 75
130  NN(G)=0
140 N.G
150 F.G=1 TO 75
160  P.G$8$8
170  X=ABSRND%75+1
180  IF NN(X)<>0 G.170
190  NN(X)=G
200 N.G
210 BE.20,15;P."  ";LI.W
220 F.X=1 TO 75
230  BE.30,5
240  K=NN(X)
250  T=K/10
260  E=K%10
270  ?#8118=T+48;?#8119=E+48
280  IF K>0  A. K<16 C=0
290  IF K>15 A. K<31 C=3
300  IF K>30 A. K<46 C=6
310  IF K>45 A. K<61 C=9
320  IF K>60 C=12
330
340  IF C=0 A. T=0 L=E
350  IF C=0 A. T=1 L=E+10
360  IF C=3 A. T=1 L=E-5
370  IF C=3 A. T=2 L=E+5
380  IF C=3 A. T=3 L=15
390  IF C=6 A. T=3 L=E
400  IF C=6 A. T=4 L=E+10
```

```
410  IF C=9 A. T=4 L=E-5
420  IF C=9 A. T=5 L=E+5
430  IF C=9 A. T=6 L=15
440  IF C=12 A. T=6 L=E
450  IF C=12 A. T=7 L=E+10
460  I=S+32*L+C;?I=#B0+T;I=I+1;?I=#B0+E
470  LI.W
480  ?I=?I-128;I=I-1;?I=?I-128
490 N.X
500 ?#E1=#80
510 E.
```

```
 10 PROGRAM BINGO 90
 20 REM J.P.TAP. UITHUIZEN
 30 REM TEL.NR. 05953-2177
 40 P.$12;?#E1=0;S=#8021;W=#FE94;@=2
 50 DIM NN(90)
 60 P."          b   i   n   g   o  "
 70 F.X=#8045 TO #805D S.3
 80  ?X=#2E
 90 N.
100 F.G=#40 TO #140 S.#20
110  F.X=#8022+G TO #803A+G S.3
120  ?X=#2E
130  N.X
140 N.G
150 F.G=1 TO 90
160  NN(G)=0
170 N.G
180 F.G=1 TO 90
190  P.G$8$8
200  X=ABSRND%90+1
210  IF NN(X)<>0 G.200
220  NN(X)=G
230 N.G
240 BE.20,15;P."  ";LI.W
250 F.X=1 TO 90
260  BE.30,5
270  K=NN(X)
280  T=K/10
290  E=K%10
300  ?#81ED=T+48;?#81EE=E+48
310  IF K>0  A. K<10 C=0
320  IF K>9  A. K<20 C=3
330  IF K>19 A. K<30 C=6
340  IF K>29 A. K<40 C=9
350  IF K>39 A. K<50 C=12
360  IF K>49 A. K<60 C=15
370  IF K>59 A. K<70 C=18
380  IF K>69 A. K<80 C=21
390  IF K>79 A. K<90 C=24
400  IF K=90 C=27
```



[illegible]

```
410  L=E+1
420  I=S+32*L+C
430  ?I=#B0+T;I=I+1;?I=#B0+E
440  LI.W
450  ?I=?I-128;I=I-1;?I=?I-128
460 N.X
470 ?#E1=#80
480 E.
```

Schakel

Inleiding

Het programma "Schakel.V2" vormt een aanvulling op de Monitor van Roel Heuvel (zie A.N. 1.7, pp.38-40 en A.N. 1.8, pp.58-59), een 1K programma gebaseerd op de 2K Apple monitor.
De uitbreiding beslaat 88 bytes machinecode.

Velen zullen zich waarschijnlijk afvragen: waarom deze uitbreiding? Waarom uberhaupt nog aandacht voor deze, ietwat verouderde, monitor annex disassembler/tracer, nu we over zoveel 'betere' debuggings-utilities beschikken, zoals de Josbox met zijn statements HDUMP, DISAS en STEP, de 6502 tracer van Bas Kasteel (zie A.N. 3.6, pp.54-55) en niet te vergeten de Program Flow Controller van Theo den Exter (zie A.N. 4.1, pp.65-75)?
Welnu, al deze utilities hebben hun specifieke voor- en nadelen: de STEP-routine uit de Josbox bijvoorbeeld, die op interruptbasis werkt, heeft als voordelen de zeer compacte weergave op het scherm en de mogelijkheid om langdradige edoch noodzakelijke initialisatie-routines snel te doorlopen. Nadelen zijn echter: het feit dat de routine op het A-blok staat, waardoor geen andere boxen geTRACEd kunnen worden, en de onmogelijkheid om er tussentijds even uit te stappen om bv. de inhoud van bepaalde geheugenlocaties te lezen: zolang de stack-pointer op #FF staat is er niets aan de hand, maar owee als je eruitstapt tijdens uitvoering van een subroutine!
De 6502 tracer van Bas Kasteel heeft eveneens zijn beperkingen: ten eerste vindt er tijdens het tracen geen disassembly plaats, waardoor een en ander zeer duister wordt, en ten tweede kunnen de registers niet tussentijds veranderd worden.
De Program Flow Controller (PFC) van Theo den Exter is ongetwijfeld de meest veelzijdige en ook de machtigste van dit drietal, zo machtig zelfs dat Theo ons afraadt om zijn debugger als monitor te 'misbruiken'... Goed, in elk geval kleeft aan zijn programma een nadeel dat ook de andere utilities kenmerkt: het gebruikt geheugen, en met name ZP-adressen... die natuurlijk net ook weer gebruikt worden door de machinetaal-routine die je wilde debuggen! Gevolg van dit "verboden access op het werkgeheugen van de debugger" (A.N. 4.1, p.73): de PFC laat het afweten.

Ook de Monitor van Roel Heuvel laat het op dit punt afweten.
Het programma heeft echter een aantal duidelijke voordelen:
- zijn compactheid: slechts 1K groot!
- zijn makkelijke hanteerbaarheid en grote flexibiliteit door de relatief eenvoudige opzet: werkt niet op interruptbasis en kan op elk moment worden onderbroken om bv. registers bij te werken of de executie elders voort te zetten.
- zijn toch vrij grote veelzijdigheid: het kan een hexdump geven, disassembleren, STEPpen, TRACEn, gegevens verplaatsen en verifieren, machinetaal-routines uitvoeren, hexadecimaal optellen en aftrekken, en dat alles met zeer eenvoudige commando's!

Kortom, een goed stuk gereedschap, ware het niet dat... juist: "verboden access", in dit geval alleen in Zero-Page (bij de PFC bv. ook op pagina #28, hetgeen in bepaalde gevallen zeer hinderlijk is).
Om nu dit laatste bezwaar tegen de Monitor weg te nemen, werd de aanvulling "Schakel.V2" geschreven.

II Werking

Wat doet deze aanvulling?

A) Zodra er een commando STEP ('S') of TRACE ('T') wordt gegeven, wordt, zowel voor als na het uitvoeren van een instructie, de hele Zero-Page verwisseld met de inhoud van een buffer-geheugen ter grootte van 256 bytes!
Hierdoor wordt het gevaar van "shared facilities", zoals Theo den Exter ze noemt, grotendeels vermeden.

Bovendien kan nu voor het buffer-geheugen een veel gunstiger gebied worden uitgezocht dan pagina #28. In "Schakel.V2" bv. is gekozen voor pagina #67 op de 16K kaart (zie de variabele "ZPSAV" in regel 190), maar dit mag natuurlijk ook een andere pagina zijn, bv. #3F of #9F.

Verder zijn in het TRACE commando enige kleine veranderingen aangebracht: ten eerste worden de opeenvolgende instructies nu door een lege regel van elkaar gescheiden, wat de leesbaarheid ten goede komt, en ten tweede wordt de TRACE-mode voortaan niet meer gestopt door de ESC-toets (wat soms vervelende gevolgen had), maar door de CTRL-toets.

Ondanks de hierboven geschetste aanvulling is de Monitor echter niet volmaakt: zo heeft het programma nog steeds een stukje geheugen nodig voor de eigenlijke uitvoering van de instructie. Hier is gekozen voor het gebied #222-#22A (zie de variabele "STAP2" in regel 170), wat echter bv. bij het tracen van een programma als CK.SOS weer problemen geeft.
Een beter alternatief is wellicht een stukje van de input-buffer, bv. #130-#138.

III Het programma

* Eerst enige opmerkingen met betrekking tot de adres-variabelen in de regels 50 t/m 220:

- r. 50 : hier staat het startadres voor de aanvullende machinecode. Als U de Monitor vast in het geheugen hebt staan dient U hiervoor een vrij stuk geheugen (88 bytes) onder battery-backup te kiezen.
- r. 80 : dit is het masker van poort B van de 8255 (#B001), dat bepaalt door welke toets de TRACE-mode wordt gestopt: voor de ESC-toets is dit #20 en voor de SHIFT-toetsen #80.
- r.100-160: de hier genoemde adressen zijn niet variabel, en mogen dus niet veranderd worden!
- r.210 : de variabele "BEGIN" bevat het beginadres van de Monitor. Deze variabele dient U aan te passen aan de plaats waar de Monitor in het geheugen van uw computer staat geassembleerd, bv. vanaf #7C00.

* In de regels 620 t/m 670 wordt de waarde van de stack-pointer gecorrigeerd, zodat bij aanroep van de Monitor een 'lege' stack-pointer wordt gesimuleerd.
  De stack-pointer kan overigens niet veranderd worden!
  Bovendien zal (in STEP- of TRACE-mode) bij de uitvoering van een subroutine de stack-pointer vertraagd worden bijgewerkt, hetgeen een enigszins vertekend beeld geeft.
  Hierover hoeft U zich echter niet ongerust te maken.

* Een laatste opmerking betreft de regels 1220 t/m 1270: deze machinecode, die zorg draagt voor correcte uitvoering van een indirecte sprong via een ZP-pointer, overschrijft de oude Monitor-routine voor het opslaan van de registers, die immers is vervangen door een nieuwe routine: regels 550 t/m 810.

IV Tot slot

Zoals reeds gezegd, ook met deze aanvulling is de Monitor van Roel Heuvel niet volmaakt.
In bepaalde debuggings-operaties zullen dan ook andere utilities, met name de PFC, de voorkeur verdienen.
Niettemin heeft de Monitor zijn aantrekkelijke kanten, in het bijzonder zijn compactheid en zijn relatieve eenvoud.
De uiteindelijke keus is, zoals altijd, aan U!



"Hou doe joe mien, kum en geddit?"

UW HOROSCOOP UIT DE ATOM.

De meesten onder u zullen zich al dan niet serieus met astrologie hebben bezig gehouden. Iedereen zal wel eens zijn horoscoop lezen zoals deze vaak in dag- en weekbladen worden gepubliceerd. Diegenen die al eens aan astrologie en het maken van horoscopen hebben edaan, weten wat een ontstellende hoeveelheid rekenwerk er aan te pas komt. Het navolgende programma is een horoscoopberekeningsprogramma. Het neemt al het vervelende rekenwerk uit handen en maakt tot slot ook nog eens de horoscooptekening.
Het programma is voor een gedeelte een vertaling van een horoscoopprogramma, welke het computertijdschrift "Computerplus" in drie afleveringen publiceerde.Het programma was oorspronkelijk bedoeld voor de Commodore 64. Bepaalde onderdelen van het programma moesten dan ook op geheel andere wijze opnieuw geprogrammeerd worden.

Over tijdstippen in een horoscoop valt heel wat te zeggen. Alles staat of valt met de invoer van de juiste tijdstippen. Er zijn dan ook een aantal tijdsoorten. Een waarnemer stelt vast dat de zon op zijn hoogste punt staat. Het is dus ter plaatse 12 uur. Op dezelfde plaats 24 uur later is het dat weer. Dit is ware ZONNETIJD: in 24 uur wordt een volledige cirkel doorlopen (=15 graden per uur). De ware zonnetijd verschilt dus van plaats tot plaats. Daarom zijn er: TIJDZONES. De nulmeridiaan (die loopt door Greenwich) geeft de wereldtijd aan. Een gebied van 7 minuten en 30 seconden (7m30s) aan beide kanten van de meridiaan heeft dezelfde tijd.
Van 7m30s tot 22m30s Oosterlengte is het 1 uur later dan in de Greenwichzone (bv. Nederland). Maar meridianen en landsgrenzen vallen niet samen. Dat is ongemakkelijk. Daarom bestaan er STANDAARDTIJDEN. Heel West-Europa heeft de middelbare Europese tijd(=Greenwichtijd plus 1 uur).
Het geboortetijdstip zoals dat geregistreerd wordt door de burgelijke stand is in MIDDELBARE PLAATSELIJKE tijd (iemand "klokt" bij de bevalling op een horloge dat de tijd aangeeft die in die tijdzone van toepassing is.) Meestal is dat dus middelbare Europese tijd.

Middelbare plaatselijke .tijd is gebaseerd op een gemiddelde; dit gemiddelde heeft twee correcties nodig in verband met afwijkingen die ontstaan door draaiingen van zon en aarde. Daarom voeren we het begrip STERRETIJD in.
De sterretijd wordt ten opzichte van een vast hemelobject (=nul graden Ram) gemeten. Om het geboortetijdstip in sterretijd uit te drukken doen we het volgende: eerst bepalen we de sterretijd op het moment dat het nul uur middelbare Greenwichtijd is op de geboortedag. Vervolgens vermenigvuldigen we de middelbare plaatselijke tijd van het tijdstip van de geboorte (decimaal genoteerd) met het getal 1.00273791 (dit is de correctie voor de bewegingen van aarde en zon). Tel de uitkomstenbij elkaar op. Dit levert het geboortetijdstip in sterretijd uitgedrukt. deze sterretijd hebben we later nodig bij het berekenen van de huizen. De Greenwichtijd is nodig voor de berekening van de

stand van de planeten.
Van belang voor het berekenen van horoscopen zijn de invoergegevens. In dit programma moeten achtereenvolgens ingevoerd worden:
NAAM - van de betreffende persoon
GEBOORTEDATUM - moet worden ingevoerd in cijfers . Het jaartal moet voluit. Tussen de cijfers moeten punten.
GEBOORTETIJDSTIP- is de werkelijke tijd van de geboorte (Kan tegen betaling worden opgevraagd bij de Burgelijke Stand)
ZOMERTIJD J/N - hier moet met J(a) of N(ee) op worden geantwoord. Verderop ziet u een lijstje waarop vermeld is in welke jaren zomertijd gold.
TIJDZONE - hier moet een getal worden ingevuld gelegen tussen -12 en +12. Voor Nederland geld +1. In het algemeen geldt dat plaatsen met een oosterlengte "+" ervoor krijgen en plaatsen met een westerlengte een "-" ervoor.
GEBOORTEPLAATS - hier moet de plaats ingevuld worden. Er zijn lengte- en breedtegraden van bepaalde plaatsen meegeprogrammeerd. Dat betekent dat men alleen de plaats hoeft in te vullen en geen antwoord meer hoeft te geven op welke lengte- en breedtegraad de plaats is gelegen.

LENGTEGRAAD - hier moet de lengtegraad in de vorm GG.MM (Graden.Minuten) worden ingevuld
BREEDTEGRAAD - hier moet de breedtegraad in de vorm GG.MM (Graden.Minuten) worden ingevuld.
Hieronder volgen enkele plaatsen in Nederland, met daarachter de breedtegraad en lengtegraad.

	Noorder-breedte	Ooster lengte
Alkmaar	52.37'	04.45'
Amsterdam	52.21'	04.55'
Assen	52.29'	06.34'
Breda	51.34'	04.48'
Eindhoven	51.26'	05.30'
Enschede	52.12'	06.53'
Den Haag	52.05'	04.18'
Groningen	53.13'	06.33'
Haarlem	52.22'	04.38'
Den Bosch	51.40'	05.20'
Hoorn	52.38'	05.04'
Leeuwarden	53.12'	05.47'
Maastricht	50.51'	05.41'
Middelburg	51.30'	03.37'
Roosendaal	51.32'	04.28'
Rotterdam	51.55'	04.30'
Tilburg	51.33'	05.07'
Utrecht	52.05'	05.08'
Wageningen	51.58'	05.40'
Zwolle	52.30'	06.05'

Aan de hand van de lengte- en breedtegraden van deze plaatsen kunt u zelf ongeveer de coordinaten van uw eigen geboorteplaats bepalen.

INITTURTLE X

Initialiseert de supersnelle turtle commando's TMOVE, RANGLE, en ANGLE. Het zal ergo deze commando's vooraf moeten gaan. De variabele heeft een waarde, overeenkomstig het CLEAR-commando, tussen 0 en 4. De turtle wordt in het midden van het scherm gezet met een hoekwaarde van 0 graden; d.w.z. naar rechts gericht.

TMOVE x,y (Turtle MOVE)

Verplaatst de turtle over het scherm; x is de plotparameter met een waarde tussen 4 en 7 (zie ook ATOM Th.&P.); y bepaalt de lengte van het lijnstuk.

ANGLE x

Hiermee zet u de turtle onder een hoek op het scherm; e is de hoekwaarde >=0; gerekend vanaf 0 graden.

RANGLE x (Relatieve ANGLE of TURN)

Verhoogt de hoekwaarde met x ten opzichte van een eerder ingenomen waarde.

Enige voorbeeld programma's

```
10 INITTURTLE 4
20 MOVE 128,96
30 RANGLE 10
40 FOR A=0 TO 2
50 TMOVE 5,70
60 RANGLE 120
70 NEXT
80 GOTO 20
```

```
 5 X=0
10 INITTURTLE 4
20 MOVE 128,96
30 TMOVE 128,0
40 FOR A=0 TO 8
50 TMOVE 5,X
60 RANGLE 33
70 NEXT A
80 X=X+1
90 GOTO 20
```

```
10 X=0
20 INITTURTLE 4
30 MOVE 128,96
40 RANGLE 4
50 FOR A=0 TO 2
60 TMOVE 5,X
70 RANGLE 120
80 NEXT A
90 X=X+2
100 GOTO 30
```

```
10 X=0;Y=0
20 INITTURTLE 4
30 MOVE 128,96
40 RANGLE 6
50 F.A=0TO7
60 TMOVE 5,X
70 RANGLE 45
80 N.A
90 Y=Y+1
100 IFY%Z=0 X=X+1
110 G.30
```

SOFTWARE: Automatische tape indexer Theo den Exter

Atomisten die BRANQUART draaien, kunnen erg aardige dingen doen met hun Atom die niet zo erg voor de hand liggen. Een handig voorbeeld daarvan wil ik presenteren in de vorm van een automatische tape indexer. Al vaker zijn er tape index programma's geschreven, maar deze hebben meestal de narigheid dat ze de index over de printer gooien, of dat je konstant bij het catalogiseren aanwezig moest zijn om de files met de hand te noteren voordat ze weer van het scherm verdwijnen.
Onder BRANQUART is het echter mogelijk om de output op te slaan via het TEXT-statement uit ED64 of WORDPACK. De tekstverwerker onthoudt dan de door de indexer samengestelde tape index. Na het catalogiseren van de tape is het aanroepen van de tekstverwerker voldoende om de gehele index te kunnen zien (om bijvoorbeeld af te drukken). Het leuke van BRANQUART is namelijk dat het FCOS en TEXT uit elkaar kan houden. De afgedrukte versie gaat van ED64 uit. Als Wordpack gewenst is, of bijvoorbeeld COS 1, dan kan dat door FCOS en/of ED64 te vervangen door COS 1 en/of EDIT. Zet de indexer wel hoog in het geheugen, zodat de wordpack file vanaf #2800 niet de indexer, of diens code overschrijft.
Deze automatische tape indexer is eigenlijk maar een voorbeeld, hoe flexibel de ATOM onder BRANQUART is.

OPMERKING van de redactie:
In de programmalisting is te zien dat in regel 680 door het programma aan het zero-page byte #FE (karakter wat niet naar de printer gestuurd wordt) geknoeid wordt. Als U het programma van Theo gebruikt verdient het aanbeveling even te kijken of de door het programma in #FE 'gepookte' waarde wel geschikt is voor de door U gebruikte printeraansturing.



EDITOR COMMANDO'S WORDPACK en ED64

(g.j.noorland)

- A (After) voegt tekst in na cursor.
  voegt buffer in met <COPY>toets direkt erna.
- B (Before) voegt tekst in voor cursor.
  voegt buffer in met <COPY>toets direkt erna.
- C (Copy) copieert tekst tussen @ en indruk <C> toets.
- D (Delete) verwijdert tekst tussen @ en indruk <D> toets.
- E (Enter) tekst wordt via de buffer in tekstgeheugen gezet.
- F (Find) zoekt tekst vanaf huidige screen.
  F/oude tekst/nieuwe tekst/a F/tekst/
  Na het zoeken antwoorden met Y of N;stoppen met <ESC>.
- H (Home) zet cursor links bovenin het scherm.
- I (Insert) voegt direkt teken in ingetypt na I voor cursor.
- N (Next) volgende scherm tekst.
- O (Option) printer aan,gevolgd door P wordt aggedrukt.
  gevolgd door andere toets komt listing,zoals afdruk wordt.
- P (Previous) vorige scherm tekst.
- Q (Quit) verlaat de tekst editor naar pagina #82.
- R (Replace) vervangt de tekst tussen @ en <R>.
  Nieuwe tekst daarna invoeren.
- S (Start) ga naar begin van de tekst.
- W (Where) geeft aan waar de tekst-endmarker zich bevindt.
- X verandert het teken boven de cursor in het direkt.
  na de <X> ingetoetste teken.
- Z ga naar de laatste deel van de tekst.
- * maakt DOS of CDS commando's mogelijk.
- > tekst naar cassette geschreven: >filenaam.
- < tekst van cassette wordt ingelezen: <filenaam.
- @ tekst marker;opheffen met <ESC>.

LET WEL:

Type na elke tekstinvoer op de <COPY> toets,i.p.v.<RETURN>.
DELETE verwijdert het teken boven de cursor en @.
Het buffergebruik gaat voor dit blad te ver,maar kan wel
Het print commando O ,geldt altijd,gevolgd door P alleen bij ED64

Wilt u lid worden van de ATOM COMPUTER CLUB.

Neem dan contact op met de penningmeester van de regio waar u bij ingedeeld wilt worden. Hij kan u vertellen, waar de regiobijeenkomsten zijn en wat u als lid kunt verwachten.

Regio NOORD:
D. Uuldriks Wiemers 14 9642 KG Veendam 05987-19611

Regio OVERIJSSEL/GELDERLAND:
H. de Ruiter Polarisstraat 25 8303 AC Emmeloord 05270-17824

Regio TWENTE:
W. Verhoeven Witbreuksweg 377-403 7522 ZA Enschede 053 -337026

Regio NOORD-HOLLAND:
P. van Kuik Zuideinde 54-a 1843 JP Groot-Schermer 02997-1902

Regio DEN HAAG:
R. Tiel Vredeoord 96 2544 TZ Den Haag 070 -294170

Regio DELFT:
P. van Alphen H. v. Delftlaan 30 2613 BN Delft 015 -122817

Regio ROTTERDAM:
R. de Haan Brasem 125 2986 HA Ridderkerk 01804-25160

Regio CENTRUM:
P. van Mourik Ruiterstede 60 3431 XN Nieuwegein 03402-48781

Regio ARNHEM:
J. Hartog Keynenbergseweg 60 6871 WK Renkum 08373-13757

Regio ZEELAND:
E. Gijssel Dorpsstraat 86 4424 CZ Wemeldinge 01192-1906

Regio BRABANT-OOST:
P. Ehrlich Roostenlaan 266 5644 BS Eindhoven 040 -114183

Regio LIMBURG:
A. v. Zandvoort Mozartstraat 328 6044 RS Roermond 04750-21797

Regio BELGIE:
R. Leyssens Oude Baan 127 3550 Heusden Belgie

Eventueel kunt u de contributie rechtstreeks overmaken per bank of giro aan de Atom Computer Club te Nuenen. Vermeld echter uw volledige naam, adres en de REGIO waarbij u ingedeeld wilt worden.